RX Family

LVD Module Using Firmware Integration Technology

Introduction

This application note describes the Voltage Detection (LVD) module which uses Firmware Integration Technology (FIT). This module uses LVD to monitor the VCC and/or an external voltage level. In this document, this module is referred to as the LVD FIT module.

Target Devices

- RX110, RX111, RX113 Groups
- RX130 Group
- RX23T, RX230, RX231 Groups
- RX23W Group
- RX24T Group
- RX24U Group
- RX64M Group
- RX65N, RX651 Group
- RX66T Group
- RX71M Group
- RX72T Group
- RX72M 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
- 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 Confirmed Operation Environment".

Contents

1. Overview

1.1 LVD FIT Module

The LVD FIT module can be used by being implemented in a project as an API. See section 2.12, Adding the FIT Module to Your Project for details on methods to implement this FIT module into a project.

1.2 Overview of the LVD FIT Module

The RX Family MCUs supported by the LVD FIT module have two channels of the LVD circuit that can be used to monitor the VCC and/or an external voltage level. Using the API functions in this module can release you from taking care of the LVD related registers that are used for low level voltage detection.

The LVD FIT module allows you to specify the voltage detection conditions and processing upon voltage detection for each channel.

- Voltage detection conditions to be specified:
  - Voltage detection level
  - Detection of the monitored voltage going above and/or below the voltage detection level.
- Processing upon voltage detection to be selected from the following:
  - Reset
  - Non-maskable interrupt
  - Maskable interrupt
  - No processing

For detailed support information for your MCU, refer to the User's Manual: Hardware for the MCU.

1.3 API Overview

Table 1.1 lists the API functions included in this module.

Table 1.1 API Functions

Function	Description
R_LVD_Open()	Initializes the specified channel and starts the LVD.
R_LVD_Close()	Stops the specified LVD channel.
R_LVD_GetStatus()	Obtains the LVD status of the specified channel.
R_LVD_ClearStatus()	Clears the voltage crossing status.
R_LVD_GetVersion()	Returns the current version of this module.

1.4 Limitations

The LVD FIT module does not support the following features:

- ELC linking

2. API Information

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

2.1 Hardware Requirements

The MCU used must support the following functions:

- LVD

2.2 Software Requirements

This driver is dependent upon the following FIT module:

- Renesas Board Support Package (r_bsp) v5.20 or higher

2.3 Supported Toolchain

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

2.4 Interrupt Vector

The voltage monitor n interrupt (n = 1, 2) is enabled by executing the R_LVD_Open function (while the macro definition LVD_CFG_ACTION_CHANNEL_n is 2 (maskable interrupt)).

Table 2.1 lists the Interrupt Vector Used in the LVD FIT Module.

Table 2.1 Interrupt Vector Used in the LVD FIT Module

Device (1)	Interrupt Vector
RX110, RX111, RX113, RX130, RX23T, RX230, RX231, RX24T, RX24U, RX64M, RX65N, RX65N_2M, RX66T, RX71M, and RX72T RX72M	LVD1 interrupt (vector no.: 88) LVD2 interrupt (vector no.: 89)
RX23W	LVD1 interrupt (vector no.: 88)

2.5 Header Files

All API calls and their supporting interface definitions are located in “r_lvd_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 option settings of this module are located in r_lvd_rx_config.h. The option names and setting values are listed in the table below:

Configuration options in r_lvd_rx_config.h	
LVD_CFG_PARAM_CHECKING_ENABLE - Default value = “BSP_CFG_PARAM_CHECKING_ENABLE”	Specifies whether to include parameter checking in the code. = 0: Omit parameter checking from the build. = 1: Include parameter checking in the build. = BSP_CFG_PARAM_CHECKING_ENABLE: Use the system default setting. Note: Code size can be reduced by excluding parameter checking from the build.
LVD_CFG_CHANNEL_1_USED 1 LVD_CFG_CHANNEL_2_USED 1	Specifies whether to use the corresponding channel. = 0: The corresponding channel is not used. = 1: The corresponding channel is used.
LVD_CFG_VDET_TARGET_CHANNEL_1 0 LVD_CFG_VDET_TARGET_CHANNEL_2 0	Specifies the target to be monitored for each channel. = 0: VCC = 1: CMPA2 pin
LVD_CFG_VOLTAGE_LEVEL_CHANNEL_1 LVD_CFG_VOLTAGE_LEVEL_CHANNEL_2 * The default value is with reference to the hardware initial value, thus it varies depending on the product used.	Specifies the voltage detection level for each channel. Set an integer value which expresses the number up to two decimal places. Example: - To set the voltage detection level to 3.00 V, specify ‘300’. - To set the voltage detection level to 4.29 V, specify ‘429’.
LVD_CFG_DIGITAL_FILTER_CHANNEL_1 0 LVD_CFG_DIGITAL_FILTER_CHANNEL_2 0	Specifies enable/disable of the digital filter for each channel. = 0: Digital filter is disabled. = 1: Digital filter is enabled.
LVD_CFG_SAMPLING_CLOCK_CHANNEL_1 LVD_CFG_SAMPLING_CLOCK_CHANNEL_2 * The default value is with reference to the hardware initial value, thus it varies depending on the product used.	With the digital filter enabled, specifies the division ratio of LOCO as the sampling clock applied to each channel. Set an integer value which expresses the division ratio. Example: - To set the division ratio to divided-by-1, specify ‘1’. - To set the division ratio to divided-by-4, specify ‘4’.

LVD_CFG_ACTION_CHANNEL_1 LVD_CFG_ACTION_CHANNEL_2 - Default value = 1	Specifies processing upon voltage detection for each channel. = 0: Reset = 1: Non-maskable interrupt = 2: Maskable interrupt = 3: No processing Note: Reset here indicates device reset. When reset is selected as processing, a reset occurs when a monitored voltage is below the voltage detection level. When reset is selected with this definition, the operation of the reset is not dependent on the voltage detection condition.
LVD_CFG_INT_PRIORITY_CHANNEL_1 LVD_CFG_INT_PRIORITY_CHANNEL_2 - Default value = 3	Specifies the interrupt priority level for each channel, with maskable interrupt selected as processing. Set the level with an integer value; setting 1 means that the priority level is the lowest level and 15 means the highest level. Example: - To set the priority level to 3, specify ‘3’. - To set the priority level to 15, specify ‘15’.
LVD_CFG_STABILIZATION_CHANNEL_1 LVD_CFG_STABILIZATION_CHANNEL_2 * The default value differs from the hardware initial value.	Specifies the reset negation timing for each channel, with reset selected as processing. = 0: After a LVD reset, negation occurs when a certain period elapses after the monitored voltage goes above the voltage detection level. = 1: Negation occurs when a certain period elapses after the LVD reset assertion. Note: “a certain period” here means a wait time after a voltage monitoring reset. Refer to the User’s Manual: Hardware for details.

2.8 Code Size

Typical code sizes 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 table lists reference values when the C compiler's compile options are set to their default values, as described in 2.3, Supported Toolchain. The compile option default values are optimization level: 2, optimization type: for size, and data endianness: little-endian. The code size varies depending on the C compiler version and compile options.

ROM and RAM code sizes

Device	Category	Memory Used						Remarks
		Renesas Compiler		GCC		IAR Compiler		
		With Parameter Checking	Without Parameter Checking	With Parameter Checking	Without Parameter Checking	With Parameter Checking	Without Parameter Checking	
RX231	ROM size (code)	1943 bytes	1783 bytes	4008 bytes	3672 bytes	3102 bytes	2834 bytes	
	RAM size	10 bytes	10 bytes	12 bytes	12 bytes	8 bytes	8 bytes	
RX23W	ROM size (code)	1639 bytes	1196 bytes	-	-	-	-	
	RAM size	5 bytes	5 bytes	-	-	-	-	
RX65N RX65N-2M	ROM size (code)	2,028 bytes	1,859 bytes	2380 bytes	2176 bytes	3323 bytes	3055 bytes	
	RAM size	10 bytes	10 bytes	12 bytes	12 bytes	10 bytes	10 bytes	
RX66T	ROM size (code)	2045 bytes	1876 bytes	4232 bytes	3904 bytes	3322 bytes	3052 bytes	
	RAM size	10 bytes	10 bytes	12 bytes	12 bytes	10 bytes	10 bytes	
RX72T	ROM size (code)	2045 bytes	1876 bytes	4232 bytes	3904 bytes	3322 bytes	3052 bytes	
	RAM size	10 bytes	10 bytes	12 bytes	12 bytes	10 bytes	10 bytes	
RX72M	ROM size (code)	2045 bytes	1876 bytes	4408 bytes	4056 bytes	3292 bytes	3020 bytes	
	RAM size	10 bytes	10 bytes	12 bytes	12 bytes	10 bytes	10 bytes	

2.9 Parameters

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

2.9.1 Channels

This enum defines channels that can be used with the MCU.

```
typedef enum
```

```
{
      LVD_CHANNEL_1 = 0,
      LVD_CHANNEL_2,
      LVD_CHANNEL_INVALID
} lvd_channel_t;
```

enum	Description
LVD_CHANNEL_1	LVD channel 1
LVD_CHANNEL_2	LVD channel 2

2.9.2 Voltage Detection Conditions

This enum defines voltage detection conditions and influences interrupt conditions and the LVD status. A reset occurs when the monitored voltage is below the voltage detection level, thus reset is not influenced by this enum.

```
typedef enum
{
      LVD_TRIGGER_RISE = 0,
      LVD_TRIGGER_FALL,
      LVD_TRIGGER_BOTH,
      LVD_TRIGGER_INVALID
} lvd_trigger_t;
```

enum	Description
LVD_TRIGGER_RISE	Rising voltage
LVD_TRIGGER_FALL	Falling voltage
LVD_TRIGGER_BOTH	Rising and falling voltages

2.9.3 Voltage Position Status

This enum defines the status, whether the monitored voltage is above or below the voltage detection level. The status is hereinafter referred to as “voltage position status” in this document.

```
typedef enum
{
      LVD_STATUS_POSITION_ABOVE = 0,
      LVD_STATUS_POSITION_BELOW,
      LVD_STATUS_POSITION_INVALID
} lvd_status_position_t;
```

enum	Description
LVD_STATUS_POSITION_ABOVE	The voltage is above the voltage detection condition.
LVD_STATUS_POSITION_BELOW	The voltage is below the voltage detection condition.

2.9.4 Voltage Crossing Status

This enum defines the status, whether the voltage crossed the voltage detection level or not. The status is hereinafter referred to as “voltage crossing status” in this document.

```
typedef enum
{
      LVD_STATUS_CROSS_NONE = 0,
      LVD_STATUS_CROSS_OVER,
      LVD_STATUS_CROSS_INVALID
} lvd_status_cross_t;
```

enum	Description
LVD_STATUS_CROSS_NONE	Not crossed the voltage detection level.
LVD_STATUS_CROSS_OVER	Crossed the voltage detection level.

2.9.5 Configuration Settings

This data structure defines the structure which is sent to the R_LVD_Open() function.

```
typedef struct
{
    lvd_trigger_t trigger;
} lvd_config_t;
```

2.9.6 Callback

This data structure defines the structure which is sent to the callback function.

```
typedef struct
{
    bsp_int_src_t vector;
} lvd_int_cb_args_t;
```

2.10 Return Values

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

```
typedef enum
{
      LVD_SUCCESS = 0,
      LVD_ERR_INVALID_PTR,
      LVD_ERR_INVALID_FUNC,
      LVD_ERR_INVALID_DATA,
      LVD_ERR_INVALID_CHAN,
      LVD_ERR_INVALID_ARG,
      LVD_ERR_UNSUPPORTED,
      LVD_ERR_ALREADY_OPEN,
      LVD_ERR_NOT_OPENED,
      LVD_ERR_LOCO_STOPPED
} lvd_err_t;
```

2.11 Callback Function

In this module, the callback function specified by the user is called when the LVD interrupt occurs.

The callback function is specified by storing the address of the user function in the “void (*p_callback)(void *)” structure member (see 2.9, Parameters). When the callback function is called, the variable which stores the constant is passed as the argument.

The callback function has an argument which is lvd_int_cb_args_t* type and the following interrupt sources are set in the vector. Refer to section 5 for the usage examples.

- BSP_INT_SRC_LVD1: LVD channel 1
- BSP_INT_SRC_LVD2: LVD channel 2

When using a value in the callback function, type cast the value.

```
void lvd_isr_callback(void *p_args)
{
  lvd_err_t                   err;
```

```
  lvd_status_position_t       status_position;
  lvd_status_cross_t          status_cross
  lvd_int_cb_args_t           *p_cb_args;

  p_cb_args = (lvd_int_cb_args_t*)p_args;
  …
}
```

2.12 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) 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 “Renesas e^2 studio Smart Configurator User Guide (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 “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 “Renesas e^2 studio Smart Configurator User Guide (R20AN0451)” 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 “Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)” for details.

2.13 “for”, “while” and “do while” statements

In this module, “for”, “while” and “do while” statements (loop processing) are used in processing to wait for register to be reflected and so on. For these loop processing, comments with “WAIT_LOOP” as a keyword are described. Therefore, if user incorporates fail-safe processing into loop processing, user can search the corresponding processing with “WAIT_LOOP”.

The following shows example of description.

```
while statement example :
/* WAIT_LOOP */
while(0 == SYSTEM.OSCOVFSR.BIT.PLOVF)
{
    /* The delay period needed is to make sure that the PLL has stabilized. */
}

for statement example :
/* Initialize reference counters to 0. */
/* WAIT_LOOP */
for (i = 0; i < BSP_REG_PROTECT_TOTAL_ITEMS; i++)
{
    g_protect_counters[i] = 0;
}

do while statement example :
/* Reset completion waiting */
do
{
    reg = phy_read(ether_channel, PHY_REG_CONTROL);
    count++;
} while ((reg & PHY_CONTROL_RESET) && (count < ETHER_CFG_PHY_DELAY_RESET)); /* WAIT_LOOP */
```

3. API Functions

R_LVD_Open

This function initializes the specified channel and starts the LVD.

Format

```
lvd_err_t       R_LVD_Open (
        lvd_channel_t                   channel,
        lvd_config_t const      *p_cfg,
        void (*p_callback)(void *)
)
```

Parameters

lvd_channel_t channel

Enumerated channel number to be initialized and for which the LVD starts.

*lvd_config_t const *p_cfg*

Address of the configuration structure.

p_callback

Address of the function which is called from an interrupt upon the voltage detection.

Return Values

```
[LVD_SUCCESS]           /* Successful: The LVD has been started.                    */
[LVD_ERR_INVALID_PTR]   /*Error: Address in the p_cfg parameter is invalid.         */
[LVD_ERR_INVALID_FUNC]  /* Error: Address in the p_callback parameter is invalid.  */
[LVD_ERR_INVALID_DATA]  /* Error: The definition of the configuration option is invalid. */
[LVD_ERR_INVALID_CHAN]  /* Error: The channel parameter is invalid.                 */
[LVD_ERR_INVALID_ARG]   /* Error: The argument of the p_cfg parameter is invalid.   */
[LVD_ERR_UNSUPPORTED]   /* Error: Selected function not supported.                  */
[LVD_ERR_ALREADY_OPEN]  /* Error: The specified channel has already been open.      */
[LVD_ERR_ LOCO_STOPPED] /* Error: Setting during LOCO is stopped is invalid.        */
```

Properties

Prototyped in file “r_lvd_rx_if.h”.

Description

This function uses the *p_cfg* parameter and the configuration option settings to initialize the specified channel and configure settings for processing upon voltage detection, and starts the LVD. When this function completes its processing successfully, the status of the channel becomes ‘Opened’.

This function is executed for each channel, however, the configuration option settings for the voltage detection level and the monitored voltage become effective only while all LVD circuits are stopped.

The callback function may or may not need to be registered in the *p_callback* parameter depending on processing upon voltage detection, which is specified in the configuration option settings. For details, see the table below.

Processing upon voltage detection (LVD_CFG_ACTION_CHANNEL_1) (LVD_CFG_ACTION_CHANNEL_2)	Necessity of callback function (Parameter: *p_callbak*)
Reset	Not needed: 'FIT_NO_FUNC' is set.
Non-maskable interrupt	Needed: Address of the callback function is set.
Maskable interrupt	Needed: Address of the callback function is set.
No processing	Not needed: 'FIT_NO_FUNC' is set.

Example

This section describes an example to specify reset as processing upon voltage detection and call this function.

For setting examples with other conditions, refer to section 5.

```
lvd_err_t   err;
lvd_config_t      cfg;

cfg.trigger = LVD_TRIGGER_FALL;
err = R_LVD_Open(LVD_CHANNEL_1, &cfg, FIT_NO_FUNC);
```

Special Notes:

The following settings of the definitions in the configuration option cannot be used while LOCO is stopped. If the following settings are specified while LOCO is stopped and this function is executed, the error 'LVD_ERR_LOCO_STOPPED' is returned.

When reset is specified as processing upon voltage detection, do not set LVD_CFG_STABILIZATION_CHANNEL_n (n = 1, 2) (reset negation timing) to 1.

Do not set LVD_CFG_DIGITAL_FILTER_CHANNEL_n (n = 1, 2) (digital filter enable/disable setting) to 1.

When a reset occurs, the LVD related registers to be initialized differ depending on the reset type. After a reset occurs, the LVD operating with registers not initialized continues operating. The configuration option settings for the voltage detection level and the monitored voltage become effective only while all LVD circuits are stopped. Thus processing such as the R_LVD_Close function needs to be performed after a reset occurs as necessary.

After a reset occurs, if the LVD operating with registers not initialized is performing processing through the software such as the callback function call by an interrupt, the processing will not be performed correctly. Processing through the software can be enabled by executing the R_LVD_Open function.

For some MCUs, LVD channel 1 and the voltage monitoring reset specified with the option function select register are combined in one function. In these MCUs, to start voltage monitoring with LVD channel 2 when the voltage monitoring reset is enabled with the option function, LVD channel 1, i.e. the voltage monitoring reset with the option function, needs to be stopped once. When restarting LVD channel 1 by this function, note that the voltage monitoring reset setting with the option function is not used but the setting in this module is used.

For details on limitations and supported functions for your MCU, refer to the User's Manual: Hardware for the MCU.

When LVD_CFG_ACTION_CHANNEL_n (n = 1, 2) is set to 1 or 2, check the voltage status of the power supply with the R_LVD_GetStatus function after executing the R_LVD_Open function, and if the status detected indicates 'LVD_STATUS_CROSS_OVER', then clear the status using the R_LVD_ClearStatus function.

R_LVD_Close

This function stops the specified LVD channel.

Format

```
lvd_err_t       R_LVD_Close (
        lvd_channel_t   channel
)
```

Parameters

lvd_channel_t channel

Enumerated channel number to be stopped.

Return Values

```
[LVD_SUCCESS]           /* Successful: The LVD has been stopped.      */
[LVD_ERR_INVALID_CHAN]  /* Error: The channel parameter is invalid.   */
```

Properties

Prototyped in file “r_lvd_rx_if.h”.

Description

This function stops the specified LVD channel. When this function completes its processing successfully, the status of the channel becomes ‘Not opened’.

Example

This section describes an example to call this function.

For setting examples with the other conditions, refer to section 5.

```
lvd_err_t err;

err = R_LVD_Close(LVD_CHANNEL_1);
```

Special Notes:

None.

R_LVD_GetStatus

This function obtains the LVD status of the specified channel.

Format

```
lvd_err_t       R_LVD_GetStatus     (
        lvd_channel_t           channel,
        lvd_status_position_t   *p_status_position,
        lvd_status_cross_t      *p_status_cross
)
```

Parameters

lvd_channel_t channel
Enumerated channel number to obtain the status.

lvd_status_position_t p_status_position
Address to store the enumerated voltage position status.

lvd_status_cross_t p_status_cross
Address to store the enumerated voltage crossing status.

Return Values

[LVD_SUCCESS]	*/* Successful: The LVD status has been obtained.*/*
[LVD_ERR_INVALID_PTR]	*/*Error: Addresses in the p_status_position and p_status_cross parameters are invalid. */*
[LVD_ERR_INVALID_CHAN]	*/* Error: The channel parameter is invalid. */*
[LVD_ERR_NOT_OPENED]	*/* Error: The specified channel is not opened. */*

Properties

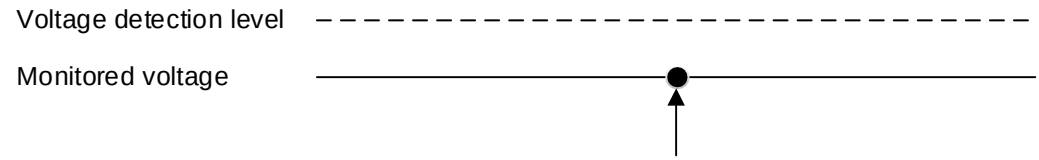
Prototyped in file “r_lvd_rx_if.h”.

Description

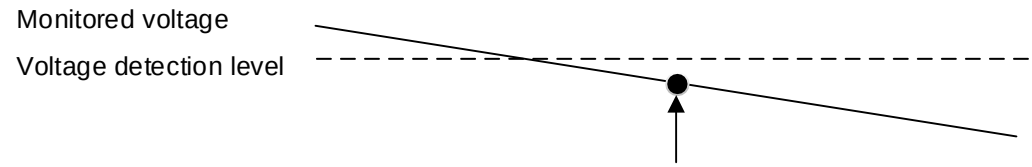
This function stores the LVD statuses into parameters **p_status_position* and **p_status_cross* for the specified channel. Refer to Figure 3.1for details on the statuses.

The voltage position status stored in the **p_status_position* parameter can be obtained without dependence on the voltage detection condition. The voltage crossing status stored in the **p_status_cross* parameter is dependent on the voltage detection condition and the status becomes ‘Crossed’ only when the condition is satisfied.

Before this function is executed, the R_LVD_Open() function must be executed with the specified channel to make the channel status ‘Opened’.



- Voltage position status: *p_status_position = LVD_STATUS_POSITION_BELOW
- Voltage crossing status: *p_status_cross = LVD_STATUS_CROSS_NONE

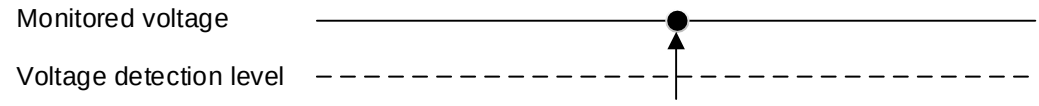


When the voltage detection condition is LVD_TRIGGER_RISE:

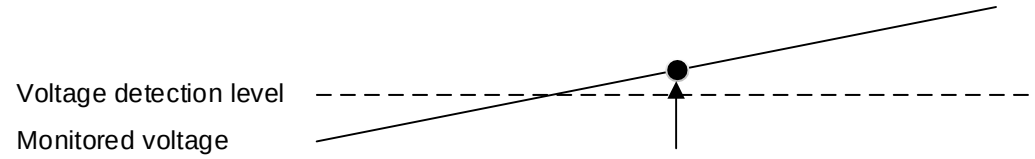
- Voltage position status: *p_status_position = LVD_STATUS_POSITION_BELOW
- Voltage crossing status: *p_status_cross = LVD_STATUS_CROSS_NONE

When the voltage detection condition is LVD_TRIGGER_FALL:

- Voltage position status: *p_status_position = LVD_STATUS_POSITION_BELOW
- Voltage crossing status: *p_status_cross = LVD_STATUS_CROSS_OVER



- Voltage positon status: *p_status_position = LVD_STATUS_POSITION_ABOVE
- Voltage crossing status: *p_status_cross = LVD_STATUS_CROSS_NONE



When the voltage detection condition is LVD_TRIGGER_RISE:

- Voltage position status: *p_status_position = LVD_STATUS_POSITION_ABOVE
- Voltage crossing status: *p_status_cross = LVD_STATUS_CROSS_OVER

When the voltage detection condition is LVD_TRIGGER_FALL:

- Voltage position status: *p_status_position = LVD_STATUS_POSITION_ABOVE
- Voltage crossing status: *p_status_cross = LVD_STATUS_CROSS_NONE

Figure 3.1 Monitored Voltage Status Relative to the Voltage Detection Level and the LVD Status

Example

This section describes an example to call this function.

For setting examples with the other conditions, refer to section 5.

```
lvd_err_t err;
lvd_status_position_t status_pos;
lvd_status_cross_t status_cross;

status = R_LVD_GetStatus (LVD_CHANNEL_1, &status_pos, &status_cross);
```

Special Notes:

None.

R_LVD_ClearStatus

This function clears the voltage crossing status for the specified channel.

Format

```
lvd_err_t       R_LVD_ClearStatus (
        lvd_channel_t   channel
)
```

Parameters

lvd_channel_t channel

Enumerated channel number to clear the voltage crossing status.

Return Values

[LVD_SUCCESS]	*/* Successful: The voltage crossing status has been cleared. */*
[LVD_ERR_INVALID_CHAN]	*/* Error: The channel parameter is invalid. */*
[LVD_ERR_NOT_OPENED]	*/* Error: The specified channel is not opened. */*

Properties

Prototyped in file “r_lvd_rx_if.h”.

Description

This function clears the voltage crossing status to ‘Not crossed’ for the specified channel. To clear the status, interrupt and reset are temporarily disabled.

Before executing this function, the R_LVD_Open() function must be executed with the specified channel to make the channel status ‘Opened’.

Example

This section describes an example to call this function.

For setting examples with the other conditions, refer to section 5.

```
lvd_err_t err;

err = R_LVD_ClearStatus (LVD_CHANNEL_1);
```

Special Notes:

Note that no interrupt or reset will occur if a voltage is detected while interrupt and reset are temporarily disabled by this function.

R_LVD_GetVersion

This function returns the current version of the LVD FIT module.

Format
uint32_t R_LVD_GetVersion (void)

Parameters
None.

Return Values
Version of this module.

Properties
Prototyped in file “r_lvd_rx_if.h”.

Description
This function returns the version of the LVD FIT module. 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.

Example
This section describes an example to call this function.

```
uint32_t version;

version = R_LVD_GetVersion();
```

Special Notes:
None.

4. Pin Setting

LVD FIT module don’t use pin setting.

5. Usage Examples

This section describes setting examples when using the LVD FIT module.

5.1 Monitoring VCC and Using Reset with LVD Channel 1

This section describes a setting example to use reset when the VCC falls to 4.29 V or lower with LVD channel 1.

Specify the following macro in the configuration option in the r_lvd_rx_config.h file.

When necessary, specify the reset negation timing and digital filter setting.

- LVD channel used: Channel 1

```
#define LVD_CFG_CHANNEL_1_USED      (1)
```

- Monitored voltage: VCC

```
#define LVD_CFG_VDET_TARGET_CHANNEL_1 (0)
```

- Voltage level: 4.29 V

```
#define LVD_CFG_VOLTAGE_LEVEL_CHANNEL_1 (429)
```

- Action: Reset

```
#define LVD_CFG_ACTION_CHANNEL_1 (0)
```

Execute the R_LVD_Open() function to start the LVD

```
void main(void)
{
      lvd_err_t err;
      lvd_config_t cfg;

      cfg.trigger = LVD_TRIGGER_FALL;
      err = R_LVD_Open(LVD_CHANNEL_1, &cfg, FIT_NO_FUNC);
}
```

5.2 Monitoring CMPA2 and Using Interrupt with LVD Channel 2

This section describes a setting example to use maskable interrupt when the CMPA2 rises to 4.29 V or higher, or when it falls to 4.29 V or lower with LVD channel 2.

Specify the following macro in the configuration option in the r_lvd_rx_config.h file.

When necessary, specify the interrupt priority level and digital filter setting.

- LVD channel used: Channel 2

```
#define LVD_CFG_CHANNEL_2_USED      (1)
```

- Monitored voltage: CMPA2

```
#define LVD_CFG_VDET_TARGET_CHANNEL_2 (1)
```

- Voltage level: 4.29 V

```
#define LVD_CFG_VOLTAGE_LEVEL_CHANNEL_2 (429)
```

- Action: Maskable interrupt

```
#define LVD_CFG_ACTION_CHANNEL_2 (2)
```

Execute the R_LVD_Open() function to start the LVD

```
void main(void)
{
      lvd_err_t err;
      lvd_config_t cfg;

      cfg.trigger = LVD_TRIGGER_BOTH;
      err = R_LVD_Open(LVD_CHANNEL_2, &cfg, (void*)lvd_isr_callback);
}
```

Prepare for the callback function as processing upon voltage detection and execute it when necessary.

```
void lvd_isr_callback(void *p_args)
{
  lvd_err_t                   err;
  lvd_status_position_t       status_position;
  lvd_status_cross_t          status_cross
  lvd_int_cb_args_t           *p_cb_args;

  p_cb_args = (lvd_int_cb_args_t*)p_args;
  if (BSP_INT_SRC_LVD2 == p_cb_args->vector)
  {
      err = R_LVD_GetStatus(LVD_CHANNEL_2, &status_position, &status_cross);
      if (status_position == LVD_STATUS_POSITION_ABOVE)
      {
            /* User code */
      }
      else
      {
            /* User code */
      }
      err = R_LVD_ClearStatus(LVD_CHANNEL_2);
  }
}
```

5.3 Obtaining the LVD Channel 2 Status

This section describes a setting example to obtain the LVD channel status with no processing upon voltage detection when CMPA2 rises to 4.29 V or higher, or when it falls to 4.29 V or lower with LVD channel 2.

Specify the following macro in the configuration option in the r_lvd_rx_config.h file.

When necessary, specify the digital filter setting.

- LVD channel used: Channel 2

```
#define LVD_CFG_CHANNEL_2_USED      (1)
```

- Monitored voltage: CMPA2

```
#define LVD_CFG_VDET_TARGET_CHANNEL_2 (1)
```

- Voltage level: 4.29 V

```
#define LVD_CFG_VOLTAGE_LEVEL_CHANNEL_2 (429)
```

- Action: No processing

```
#define LVD_CFG_ACTION_CHANNEL_2 (3)
```

Execute the R_LVD_Open() function to start the LVD and obtain the status.

```
void main(void)
{
  lvd_err_t                   err;
  lvd_config_t                cfg;
  lvd_status_position_t       status_pos;
  lvd_status_cross_t          status_cross;

  cfg.trigger = LVD_TRIGGER_BOTH;
  err = R_LVD_Open(LVD_CHANNEL_2, &cfg, FIT_NO_FUNC);

  err = R_LVD_GetStatus (LVD_CHANNEL_2, &status_pos, &status_cross);
}
```

5.4 Changing the Voltage Detection Condition with LVD Channel 1

This section describes a setting example to change the voltage detection condition when the VCC rises to 4.29 V or higher after the LVD has started while the maskable interrupt is used when the VCC falls to 4.29 V or lower with LVD channel 1.
Specify the following macro in the configuration option in the r_lvd_rx_config.h file.
When necessary, specify reset negation timing and the digital filter setting.

- LVD channel used: Channel 1

```
#define LVD_CFG_CHANNEL_1_USED      (1)
```

- Monitored voltage: VCC

```
#define LVD_CFG_VDET_TARGET_CHANNEL_1 (0)
```

- Voltage level: 4.29 V

```
#define LVD_CFG_VOLTAGE_LEVEL_CHANNEL_1 (429)
```

- Action: Maskable interrupt

```
#define LVD_CFG_ACTION_CHANNEL_1 (2)
```

Start the LVD with the R_LVD_Open() function, change the voltage detection condition, and then restart the LVD.

```
void main(void)
{
  lvd_err_t err;
  lvd_config_t cfg;

  cfg.trigger = LVD_TRIGGER_FALL;
  err = R_LVD_Open(LVD_CHANNEL_1, &cfg, (void*)lvd_isr_callback);

  err = R_LVD_Close(LVD_CHANNEL_1);

  cfg.trigger = LVD_TRIGGER_RISE;
  err = R_LVD_Open(LVD_CHANNEL_1, &cfg, (void*)lvd_isr_callback);
}
```

Prepare for the callback function for processing upon voltage detection and execute it when necessary.

```
void lvd_isr_callback(void *p_args)
{
  lvd_err_t                   err;
  lvd_status_position_t       status_position;
  lvd_status_cross_t          status_cross
  lvd_int_cb_args_t           *p_cb_args;

  p_cb_args = (lvd_int_cb_args_t*)p_args;
  if (BSP_INT_SRC_LVD1 == p_cb_args->vector)
  {
      err = R_LVD_GetStatus(LVD_CHANNEL_1, &status_position, &status_cross);
      if (status_position == LVD_STATUS_POSITION_ABOVE)
      {
            /* User code */
      }
      else
      {
            /* User code */
      }
      err = R_LVD_ClearStatus(LVD_CHANNEL_1);
  }
}
```

6. Demo Projects

Demo projects include function main() that utilizes the FIT module and its dependent modules (e.g. r_bsp). This FIT module includes the following demo projects.

6.1 lvd_demo_rskrx113

This section describes the operating conditions for lvd_demo_rskrx113. The demo project demonstrates how to configure the callback function which uses the LVD interrupt and refer the channel information of the callback parameter. In the program, with the LVD callback function, LED0 is turned on when detected the voltage is rising and LED0 is turned off when detected the voltage is falling. LED1 is turned on while the demo project is operating to indicate the program is being executed.

- LVD channel used: Channel 1
- Monitored voltage: VCC
- Voltage level: 2.90 V
- Action: Non-maskable interrupt
- Voltage detection condition: Rising voltage or falling voltage

Note:

- LVD demo should be run in stand-alone mode (download program to board, unplug debugger, supply external power).
- Need to use a variable power supply unit (PSU) to adjust the input voltage. Plug this PSU to power jack (PWR) on RSK board.

6.2 lvd_demo_rskrx231

The lvd_demo_rskrx231 program is identical to lvd_demo_rskrx113.

6.3 lvd_demo_rskrx64m

The lvd_demo_rskrx64m program is identical to lvd_demo_rskrx113 except for the voltage level.

- Voltage level: 2.99 V

6.4 lvd_demo_rskrx65n

The lvd_demo_rskrx65n program is identical to lvd_demo_rskrx113 except for the voltage level.

- Voltage level: 2.85 V

6.5 lvd_demo_rskrx65n_2m

The lvd_demo_rskrx65n_2m program is identical to lvd_demo_rskrx113 except for the voltage level.

- Voltage level: 2.85 V

6.6 Adding a Demo to a Workspace

Demo projects are found in the FITDemos 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 FITDemos subdirectory, select the desired demo zip file, then click “Finish”.

6.7 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 this application note and select “Sample Code (download)” from the context menu in the *Smart Browser* >> *Application Notes* tab.

7. Appendices

7.1 Confirmed Operation Environment

This section describes confirmed operation environment for the LVD FIT module.

Table 7.1 Confirmed Operation Environment (Rev.3.20)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.5.0 IAR Embedded Workbench for Renesas RX 4.12.1
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
	GCC for Renesas RX 4.8.4.201902 Compiler option: The following option is added to the default settings of the integrated development environment. -std=gnu99 Linker option: The following user defined option should be added to the default settings of the integrated development environment, if “Optimize size (-Os)” is used: -Wl,--no-gc-sections This is to work around a GCC linker issue whereby the linker erroneously discard interrupt functions declared in FIT peripheral module
	IAR C/C++ Compiler for Renesas RX version 4.12.1 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.3.20
Board used	Renesas Starter Kit+ for RX72M (product No.: RTK5572Mxxxxxxxxxx)

Table 7.2 Confirmed Operation Environment (Rev.3.10)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.5.0
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.3.10
Board used	Renesas Solution Starter Kit for RX23W (product No.: RTK5523Wxxxxxxxxxx)

Table 7.3 Confirmed Operation Environment (Rev.3.00)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.4.0 IAR Embedded Workbench for Renesas RX 4.10.1
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
	GCC for Renesas RX 4.8.4.201803 Compiler option: The following option is added to the default settings of the integrated development environment.

	-std=gnu99 Linker option: The following user defined option should be added to the default settings of the integrated development environment, if “Optimize size (-Os)” is used: -Wl,--no-gc-sections This is to work around a GCC linker issue whereby the linker erroneously discard interrupt functions declared in FIT peripheral module
	IAR C/C++ Compiler for Renesas RX version 4.10.1 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 RX65N-2MB (product No.: RTK50565Nxxxxxxxxx)

Table 7.4 Confirmed Operation Environment (Rev.2.50)

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.2.50
Board used	Renesas Starter Kit for RX72T (product No.: RTK5572Txxxxxxxxxx)

Table 7.5 Confirmed Operation Environment (Rev.2.41)

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.2.41
Board used	Renesas Starter Kit for RX66T (product No.: RTK50566T0SxxxxxBE) Renesas Starter Kit+ for RX 65N-2MB (product No.: RTK50565N2SxxxxxBE) Renesas Starter Kit+ for RX130-512KB (product No.: RTK5051308SxxxxxBE)

Table 7.6 Confirmed Operation Environment (Rev.2.40)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.0.0.
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.00.00
	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.2.40
Board used	Renesas Starter Kit for RX66T (product No.: RTK50566T0SxxxxxBE) Renesas Starter Kit+ for RX 65N-2MB (product No.: RTK50565N2SxxxxxBE) Renesas Starter Kit+ for RX130-512KB (product No.: RTK5051308SxxxxxBE)

Table 7.7 Confirmed Operation Environment (Rev.2.31)

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.2.31
Board used	Renesas Starter Kit+ for RX 65N-2MB (product No.: RTK50565N2SxxxxxBE) Renesas Starter Kit+ for RX130-512KB (product No.: RTK5051308SxxxxxBE)

Table 7.8 Confirmed Operation Environment (Rev.2.30)

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.2.30
Board used	Renesas Starter Kit+ for RX 65N-2MB (product No.: RTK50565N2SxxxxxBE) Renesas Starter Kit+ for RX130-512KB (product No.: RTK5051308SxxxxxBE)

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 this 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_lvd_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_lvd_rx_config.h” may be wrong. Check the file “r_lvd_rx_config.h”. If there is a wrong setting, set the correct value for that. Refer to 2.7, Configuration Overview for details.

8. Reference Documents

User's Manual: Hardware
The latest versions 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 CC-RX User's Manual (R20UT3248)
The latest version can be downloaded from the Renesas Electronics website.

Related Technical Updates

This module reflects the content of the following technical updates.

TN-RX*-A137A/E

RX230 Group, RX231 Group User's Manual: Hardware Rev.1.00 (Page 1941 of 1968)

Value in Table 50.57, Characteristics of Power-On Reset Circuit and Voltage Detection Circuit (1) is corrected.

Revision History

		Description	
Rev.	Date	Page	Summary
2.00	June 15, 2016	—	First Release.
2.10	Oct. 01, 2016	—	Added support for the RX65N Group.
		1	Added a note; description of the compatibility with “RX Family LVD Module Using Firmware Integration Technology (R01AN1726)”.
2.20	Feb. 28, 2017	—	Added support for the RX24U Group.
		—	Corrected a description.
		4	Added RXC v2.06.00 to “2.5 Supported Toolchains”.
		12	Added a note for when an interrupt is enabled in the Special Notes in 3.2 R_LVD_Open.
		Program	The code has been modified to check arguments for NULL, FIT_NO_PTR, and FIT_NO_FUNC.
2.30	July 24, 2017	—	Added support for RX130-512KB and RX65N-2MB.
		1	Related Documents: Added the following document: “Renesas e^2 studio Smart Configurator User Guide (R20AN0451)”
		4	2.6 Interrupt Vector: Added.
		9	2.14 Adding the FIT Module to Your Project: Revised.
		23	5.5 Downloading Demo Projects
		24-25	6. Appendices: Added.
2.31	Oct. 31, 2017	—	Added support for RX65N and RX65N-2MB.
		4	2.6 Interrupt Vector: Added RX65N and RX65N-2MB.
		6	2.10 Code Size: Add code size corresponding to RX65N, RX65N_2M
		23	5.1 lvd_demo_rskrx113: Added additional operating condition
		23	5.4 lvd_demo_rskrx65n: Added
		23	5.5 lvd_demo_rskrx65n_2m: Added
		24	6.1 Operation Confirmation Environment: Added Table for Rev. 2.31
		25	6.2 Troubleshooting: Added one more question.
2.40	Sep 28, 2018	1, 5	Added support for the RX66T.
		8	Added code size corresponding to RX66T
		28	7.1 Confirmed Operation Environment: Added Table for Rev.2.40
2.41	Nov 16, 2018	—	Added document number in XML
		1	Added support for the RX651
		28	Changed Renesas Starter Kit Product No for RX66T. Added Table for Rev.2.41
2.50	Feb 01, 2019	—	Fixed bug in MDF file which causes abnormal behavior in Smart Configurator when setting the “voltage detection level”
		Program	Added support for RX72T.
		1, 5	Added support for RX72T.
		8	Added code size corresponding to RX72T
		14-20	Removed ‘Reentrant’ description in each API function.
		28	7.1 Confirmed Operation Environment: Added Table for Rev 2.50

3.00	May.20.19	—	Supported the following compilers: - GCC for Renesas RX - IAR C/C++ Compiler for Renesas RX
		1	Deleted the RX631 and RX63N in Target Devices for end of update these devices. Added the section of Target compilers. Deleted related documents.
		5	2.2 Software Requirements Requires r_bsp v5.20 or higher
		8	Updated the section of 2.8 Code Size
		28	Table 7.1 Confirmed Operation Environment: Added table for Rev.3.00
		31	Deleted the section of Website and Support.
		Program	Changed below for support GCC and IAR compiler: 1. Deleted the inline expansion of the R_LVD_GetVersion function. 2. Replaced the declaration of interrupt functions with the macro definition of BSP.
3.10	Jun.28.19	1, 5	Added support for RX23W
		8	Added code size corresponding to RX23W
		28	7.1 Confirmed Operation Environment: Added Table for Rev.3.10
		Program	Added support for RX23W.
3.20	Aug.15.19	1, 5	Added support for RX72M
		8	Added code size corresponding to RX72M
		27	7.1 Confirmed Operation Environment: Added Table for Rev.3.20 Table 7.2: Corrected board name for RX23W
		Program	Added support for RX72M.

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

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/.

Trademarks

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