APPLICATION NOTE

RX Family

R01AN1721EJ0210
Rev. 2.10
Oct. 01, 2016

GPIO Module Using Firmware Integration Technology

Introduction

This Firmware Integration Technology (FIT) Module implements a General Purpose Input/Output Driver.

Target Device

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

- **RX110 Group**
- **RX111 Group**
- **RX113 Group**
- **RX130 Group**
- **RX210 Group**
- **RX230 Group**
- **RX231 Group**
- **RX23T Group**
- **RX24T Group**
- **RX63N Group**
- **RX64M Group**
- **RX65N Group**
- **RX71M Group**

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

Related Documents

- Firmware Integration Technology User’s Manual (R01AN1833)
- Board Support Package Firmware Integration Technology Module (R01AN1685)
- Adding Firmware Integration Technology Modules to Projects (R01AN1723)
- Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)

Contents

1. Overview

This module provides an abstraction layer for reading, writing, and configuring General Purpose Input/Output (GPIO) pins on RX MCUs. By using this module’s API functions the user can bypass the need to know of the available GPIO registers for each pin. For example, on the RX there are separate registers for controlling pin direction, reading the pin, writing to the pin, enabling internal pull-ups, configuring output mode, and assigning a pin for peripheral use.

2. API Information

The sample code in this application note has been run and confirmed under the following conditions.

2.1 Hardware Requirements

This driver requires that your MCU supports the following peripheral(s):

- GPIO pins that can be read, written, and set as inputs or outputs.

2.2 Hardware Resource Requirements

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

2.2.1 GPIO Registers

This module's API functions have the ability to write all GPIO registers (e.g. port direction, read, write). Also the user can modify these registers in their user code.

2.3 Software Requirements

This driver is dependent upon the following FIT packages:

- Renesas Board Support Package (r_bsp)

2.4 Limitations

Some MCU packages with reduced pin counts have a Port Switching feature that multiplexes some I/O ports to shared physical pins. This driver does not support the Port Switching feature. Port Switching may still be done outside of this driver by the user's own code.

2.5 Supported Toolchains

This driver is tested and working with the following toolchains:

- Renesas RX Toolchain v.2.02.00 (RX110, RX111, RX113, RX210, RX231, RX63N, RX64M, RX71M)
- Renesas RX Toolchain v.2.03.00 (RX130, RX230, RX23T, RX24T)
- Renesas RX Toolchain v.2.05.00 (RX65N)

2.6 Header Files

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

Build-time configuration options are selected or defined in the file "r_gpio_rx_config.h".

Both of these files should be included by the user's application.

2.7 Integer Types

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.

2.8 Configuration Overview

This driver is configured using the *r_gpio_rx_config.h* header file.

All configurable options that can be set at build time are located in the file "r_gpio_rx_config.h". A summary of these settings are provided in Table 1.

Configuration options in r_gpio_rx_config.h	
Equate	**Description**
`GPIO_CFG_PARAM_CHECKING_ENABLE`	= 1: Include parameter checking in the build. = 0: Omit parameter checking from the build. = BSP_CFG_PARAM_CHECKING_ENABLE (default): Use the system default setting. Note: Code size can be reduced by excluding parameter checking from the build.

Table 1: Info about the configuration

2.9 Code Size

The code size is based on optimization level 2 and optimization type for size for the RXC toolchain in Section 2.5. The ROM (code and constants) and RAM (global data) sizes are determined by the build-time configuration options set in the module configuration header file.

ROM and RAM code sizes		
	With Parameter Checking	**Without Parameter Checking**
RX110	ROM: 511 bytes code	ROM: 355 bytes code
	RAM: 0 bytes	RAM: 0 bytes
RX111, RX113	ROM: 514 bytes code	ROM: 355 bytes code
	RAM: 0 bytes	RAM: 0 bytes
RX130, RX230	ROM: 613 bytes code	ROM: 441 bytes code
	RAM: 0 bytes	RAM: 0 bytes
RX210	ROM: 621 bytes code	ROM: 428 bytes code
	RAM: 0 bytes	RAM: 0 bytes
RX231 RX64M, RX71M	ROM: 613 bytes code	ROM: 428 bytes code
	RAM: 0 bytes	RAM: 0 bytes
RX23T, RX24T	ROM: 597 bytes code	ROM: 441 bytes code
	RAM: 0 bytes	RAM: 0 bytes
RX65N	ROM: 707 bytes code	ROM: 495 bytes code
	RAM: 0 bytes	RAM: 0 bytes

Table 2: ROM and RAM code size

2.10 API Data Types

This section details the enumerators that are used with the driver's API functions.

2.10.1 Ports

This enum defines the available ports for this MCU. This enum is specific to a MCU Group and package and is stored in the target folder for that MCU Group. For example, to find the available ports for an RX111 the user would refer to the file *src/targets/rx111/r_gpio_rx111.h*. Below is an example from the RX111. Other MCUs will have different ports enumerations.

```
#if   (BSP_PACKAGE_PINS == 64)
typedef enum
{
    GPIO_PORT_0 = 0x0000,
    GPIO_PORT_1 = 0x0100,
    GPIO_PORT_2 = 0x0200,
    GPIO_PORT_3 = 0x0300,
    GPIO_PORT_4 = 0x0400,
    GPIO_PORT_5 = 0x0500,
    GPIO_PORT_A = 0x0A00,
    GPIO_PORT_B = 0x0B00,
    GPIO_PORT_C = 0x0C00,
    GPIO_PORT_E = 0x0E00,
    GPIO_PORT_H = 0x1100,
    GPIO_PORT_J = 0x1200,
} gpio_port_t;
```

2.10.2 Pins

This enum defines the available GPIO pins for this MCU. This enum is specific to an MCU Group and package. The user can find this enum in the target folder for that MCU Group. For example, to find the available GPIO pins for an RX111 the user would refer to the file *src/targets/rx111/r_gpio_rx111.h*. Below is an example from the RX111. Notice that the GPIO pins available in this enum are controlled by the BSP_PACKAGE_PINS macro which is automatically obtained from the r_bsp.

```
#if   (BSP_PACKAGE_PINS == 64)
typedef enum
{
    GPIO_PORT_0_PIN_3 = 0x0003,
    GPIO_PORT_0_PIN_5 = 0x0005,
    GPIO_PORT_1_PIN_4 = 0x0104,
    GPIO_PORT_1_PIN_5 = 0x0105,
    GPIO_PORT_1_PIN_6 = 0x0106,
    GPIO_PORT_1_PIN_7 = 0x0107,
    GPIO_PORT_2_PIN_6 = 0x0206,
    GPIO_PORT_2_PIN_7 = 0x0207,
    GPIO_PORT_3_PIN_0 = 0x0300,
    GPIO_PORT_3_PIN_1 = 0x0301,
    GPIO_PORT_3_PIN_2 = 0x0302,
    GPIO_PORT_3_PIN_5 = 0x0305,
    GPIO_PORT_4_PIN_0 = 0x0400,
    GPIO_PORT_4_PIN_1 = 0x0401,
    GPIO_PORT_4_PIN_2 = 0x0402,
    GPIO_PORT_4_PIN_3 = 0x0403,
    GPIO_PORT_4_PIN_4 = 0x0404,
    GPIO_PORT_4_PIN_6 = 0x0406,
    GPIO_PORT_5_PIN_4 = 0x0504,
    GPIO_PORT_5_PIN_5 = 0x0505,
    GPIO_PORT_A_PIN_0 = 0x0A00,
    GPIO_PORT_A_PIN_1 = 0x0A01,
    GPIO_PORT_A_PIN_3 = 0x0A03,
    GPIO_PORT_A_PIN_4 = 0x0A04,
    GPIO_PORT_A_PIN_6 = 0x0A06,
    GPIO_PORT_B_PIN_0 = 0x0B00,
    GPIO_PORT_B_PIN_1 = 0x0B01,
    GPIO_PORT_B_PIN_3 = 0x0B03,
    GPIO_PORT_B_PIN_5 = 0x0B05,
    GPIO_PORT_B_PIN_6 = 0x0B06,
    GPIO_PORT_B_PIN_7 = 0x0B07,
    GPIO_PORT_C_PIN_0 = 0x0C00,
    GPIO_PORT_C_PIN_1 = 0x0C01,
    GPIO_PORT_C_PIN_2 = 0x0C02,
    GPIO_PORT_C_PIN_3 = 0x0C03,
    GPIO_PORT_C_PIN_4 = 0x0C04,
    GPIO_PORT_C_PIN_5 = 0x0C05,
    GPIO_PORT_C_PIN_6 = 0x0C06,
    GPIO_PORT_C_PIN_7 = 0x0C07,
    GPIO_PORT_E_PIN_0 = 0x0E00,
    GPIO_PORT_E_PIN_1 = 0x0E01,
    GPIO_PORT_E_PIN_2 = 0x0E02,
    GPIO_PORT_E_PIN_3 = 0x0E03,
    GPIO_PORT_E_PIN_4 = 0x0E04,
    GPIO_PORT_E_PIN_5 = 0x0E05,
    GPIO_PORT_E_PIN_6 = 0x0E06,
    GPIO_PORT_E_PIN_7 = 0x0E07,
    GPIO_PORT_H_PIN_7 = 0x1107,
    GPIO_PORT_J_PIN_6 = 0x1206,
    GPIO_PORT_J_PIN_7 = 0x1207,
} gpio_port_pin_t;

#elif (BSP_PACKAGE_PINS == 48)
typedef enum
{
    GPIO_PORT_1_PIN_4 = 0x0104,
    GPIO_PORT_1_PIN_5 = 0x0105,
    GPIO_PORT_1_PIN_6 = 0x0106,
    GPIO_PORT_1_PIN_7 = 0x0107,
    GPIO_PORT_2_PIN_6 = 0x0206,
    GPIO_PORT_2_PIN_7 = 0x0207,
    GPIO_PORT_3_PIN_5 = 0x0305,
    GPIO_PORT_4_PIN_0 = 0x0400,
    GPIO_PORT_4_PIN_1 = 0x0401,
    GPIO_PORT_4_PIN_2 = 0x0402,
    GPIO_PORT_4_PIN_6 = 0x0406,
    GPIO_PORT_A_PIN_1 = 0x0A01,
    GPIO_PORT_A_PIN_3 = 0x0A03,
    GPIO_PORT_A_PIN_4 = 0x0A04,
    GPIO_PORT_A_PIN_6 = 0x0A06,
    GPIO_PORT_B_PIN_0 = 0x0B00,
    GPIO_PORT_B_PIN_1 = 0x0B01,
    GPIO_PORT_B_PIN_3 = 0x0B03,
    GPIO_PORT_B_PIN_5 = 0x0B05,
    GPIO_PORT_C_PIN_0 = 0x0C00,
    GPIO_PORT_C_PIN_1 = 0x0C01,
    GPIO_PORT_C_PIN_2 = 0x0C02,
    GPIO_PORT_C_PIN_3 = 0x0C03,
    GPIO_PORT_C_PIN_4 = 0x0C04,
    GPIO_PORT_C_PIN_5 = 0x0C05,
    GPIO_PORT_C_PIN_6 = 0x0C06,
    GPIO_PORT_C_PIN_7 = 0x0C07,
    GPIO_PORT_E_PIN_0 = 0x0E00,
    GPIO_PORT_E_PIN_1 = 0x0E01,
    GPIO_PORT_E_PIN_2 = 0x0E02,
    GPIO_PORT_E_PIN_3 = 0x0E03,
    GPIO_PORT_E_PIN_4 = 0x0E04,
    GPIO_PORT_E_PIN_7 = 0x0E07,
    GPIO_PORT_H_PIN_7 = 0x1107,
    GPIO_PORT_J_PIN_6 = 0x1206,
    GPIO_PORT_J_PIN_7 = 0x1207,
} gpio_port_pin_t;

#elif (BSP_PACKAGE_PINS == 40)
typedef enum
{
    GPIO_PORT_1_PIN_4 = 0x0104,
    GPIO_PORT_1_PIN_5 = 0x0105,
    GPIO_PORT_1_PIN_6 = 0x0106,
    GPIO_PORT_1_PIN_7 = 0x0107,
    GPIO_PORT_2_PIN_6 = 0x0206,
    GPIO_PORT_2_PIN_7 = 0x0207,
    GPIO_PORT_3_PIN_2 = 0x0302,
    GPIO_PORT_3_PIN_5 = 0x0305,
    GPIO_PORT_4_PIN_1 = 0x0401,
    GPIO_PORT_4_PIN_2 = 0x0402,
    GPIO_PORT_4_PIN_6 = 0x0406,
    GPIO_PORT_A_PIN_1 = 0x0A01,
    GPIO_PORT_A_PIN_3 = 0x0A03,
    GPIO_PORT_A_PIN_4 = 0x0A04,
    GPIO_PORT_A_PIN_6 = 0x0A06,
    GPIO_PORT_B_PIN_0 = 0x0B00,
    GPIO_PORT_B_PIN_3 = 0x0B03,
    GPIO_PORT_C_PIN_4 = 0x0C04,
    GPIO_PORT_E_PIN_0 = 0x0E00,
    GPIO_PORT_E_PIN_1 = 0x0E01,
    GPIO_PORT_E_PIN_2 = 0x0E02,
    GPIO_PORT_E_PIN_3 = 0x0E03,
    GPIO_PORT_E_PIN_4 = 0x0E04,
    GPIO_PORT_J_PIN_6 = 0x1306,
    GPIO_PORT_J_PIN_7 = 0x1307,
} gpio_port_pin_t;

#elif (BSP_PACKAGE_PINS == 36)
typedef enum
{
    GPIO_PORT_1_PIN_4 = 0x0104,
    GPIO_PORT_1_PIN_5 = 0x0105,
    GPIO_PORT_1_PIN_6 = 0x0106,
    GPIO_PORT_1_PIN_7 = 0x0107,
    GPIO_PORT_2_PIN_7 = 0x0207,
    GPIO_PORT_3_PIN_5 = 0x0305,
    GPIO_PORT_4_PIN_1 = 0x0401,
    GPIO_PORT_4_PIN_2 = 0x0402,
    GPIO_PORT_A_PIN_3 = 0x0A03,
    GPIO_PORT_A_PIN_4 = 0x0A04,
    GPIO_PORT_A_PIN_6 = 0x0A06,
    GPIO_PORT_B_PIN_0 = 0x0B00,
    GPIO_PORT_B_PIN_3 = 0x0B03,
    GPIO_PORT_C_PIN_4 = 0x0C04,
    GPIO_PORT_E_PIN_0 = 0x0E00,
    GPIO_PORT_E_PIN_1 = 0x0E01,
    GPIO_PORT_E_PIN_2 = 0x0E02,
    GPIO_PORT_E_PIN_3 = 0x0E03,
    GPIO_PORT_E_PIN_4 = 0x0E04,
    GPIO_PORT_J_PIN_6 = 0x1306,
    GPIO_PORT_J_PIN_7 = 0x1307
} gpio_port_pin_t;
#endif
```

2.10.3 Port_Pin Masks

This enum is specific to the MCU Group and package. It defines port-pin masks for this MCU. These are bit-masks that may optionally be applied by the user when performing port-wide writes or reads to check for valid port-wide operations. For each bit location in a port that has an I/O pin available, these bit-masks will have the bit set to '1'. Bits for non-existent pins are set to '0'. In the interest of performance, the GPIO driver does not automatically check for invalid pin settings when the port-wide write function is called. It is up to the user's application to insure that only valid pins are written to. Use of these masks is not required; they are provided as a convenience. Below is an example from the RX111 MCU.

```
#if (BSP_PACKAGE_PINS == 64)
/* This enumerator has a bit mask for each available GPIO pin for the given port
on this MCU. */
typedef enum
{
   GPIO_PORT0_PIN_MASK = 0x28,   /* Available pins: P03,P05                  */
   GPIO_PORT1_PIN_MASK = 0xF0,   /* Available pins: P14, P15, P16,P17        */
   GPIO_PORT2_PIN_MASK = 0xC0,   /* Available pins: P26,P27                  */
   GPIO_PORT3_PIN_MASK = 0x27,   /* Available pins: P30 to P32, P35          */
   GPIO_PORT4_PIN_MASK = 0x5F,   /* Available pins: P40 to P44, P46          */
   GPIO_PORT5_PIN_MASK = 0x30,   /* Available pins: P54, P55                 */
   GPIO_PORTA_PIN_MASK = 0x5B,   /* Available pins: PA0, PA1, PA3, PA4, PA6  */
   GPIO_PORTB_PIN_MASK = 0xEB,   /* Available pins: PB0, PB1, PB3, PB5 to PB7*/
   GPIO_PORTC_PIN_MASK = 0xFF,   /* Available pins: PC0 to PC7               */
   GPIO_PORTE_PIN_MASK = 0xFF,   /* Available pins: PE0 to PE7               */
   GPIO_PORTH_PIN_MASK = 0x80,   /* Available pins: PH7                      */
   GPIO_PORTJ_PIN_MASK = 0xC0    /* Available pins: PJ6, PJ7                 */
} gpio_pin_bit_mask_t;
```

2.10.4 Pin Level

This enum defines the different options that can be returned when reading a GPIO pin.

```
/* Levels that can be set and read for individual pins. */
typedef enum
{
    GPIO_LEVEL_LOW = 0,
    GPIO_LEVEL_HIGH
} gpio_level_t;
```

2.10.5 Pin Direction

This enum defines the different options that can be used for configuring a GPIO pin's direction.

```
/* Options that can be used with the R_GPIO_PortDirectionSet() and
   R_GPIO_PinDirectionSet() functions. */
typedef enum
{
    GPIO_DIRECTION_INPUT = 0,
    GPIO_DIRECTION_OUTPUT
} gpio_dir_t;
```

2.10.6 Control Commands

This enum defines the different commands that can be sent to the R_GPIO_PinControl() function.

```
/* Commands that can be used with the R_GPIO_PinControl() function. This list
   will vary depending on the MCU chosen. */
typedef enum
{
    GPIO_CMD_OUT_CMOS = 0,
    GPIO_CMD_OUT_OPEN_DRAIN_N_CHAN,
    GPIO_CMD_OUT_OPEN_DRAIN_P_CHAN,
    GPIO_CMD_IN_PULL_UP_DISABLE,
    GPIO_CMD_IN_PULL_UP_ENABLE,
    GPIO_CMD_ASSIGN_TO_PERIPHERAL,
    GPIO_CMD_ASSIGN_TO_GPIO,
    GPIO_CMD_DSCR_DISABLE,
    GPIO_CMD_DSCR_ENABLE,
    GPIO_CMD_DSCR2_DISABLE,
    GPIO_CMD_DSCR2_ENABLE
} gpio_cmd_t;
```

2.11 Return Values

Below are the available return values for the R_GPIO_PinControl() function.

```
/* Function return type. */
typedef enum
{
    GPIO_SUCCESS = 0,
    GPIO_ERR_INVALID_MODE,  // The mode specified cannot be applied to this pin
    GPIO_ERR_INVALID_CMD    // The input command is not supported
} gpio_err_t;
```

2.12 Adding a FIT Module to Your Project

The FIT module must be added to each project in the e^2 studio.

You can use the FIT plug-in to add the FIT module to your project, or the module can be added manually.

It is recommended to use the FIT plug-in as you can add the module to your project easily and also it will automatically update the include file paths for you.

To add the FIT module using the plug-in, refer to chapter 2. “Adding FIT Modules to e^2 studio Projects Using FIT Plug-In” in the “Adding Firmware Integration Technology Modules to Projects” application note (R01AN1723).

To add the FIT module manually, refer to chapter 3. “Adding FIT Modules to e^2 studio Projects Manually” in the “Adding Firmware Integration Technology Modules to Projects (R01AN1723)”

When using the FIT module, the BSP FIT module also needs to be added. For details on the BSP FIT module, refer to the “Board Support Package Module Using Firmware Integration Technology” application note (R01AN1685).

3. API Functions

3.1 Summary

The following functions are included in this design:

Function	Description
R_GPIO_PortWrite()	Sets the output levels of all pins on a port.
R_GPIO_PortRead()	Reads the current levels of all pins on a port.
R_GPIO_PortDirectionSet()	Sets multiple pins on a port as inputs or outputs.
R_GPIO_PinWrite()	Sets the output level of a pin.
R_GPIO_PinRead()	Reads the current level of a pin.
R_GPIO_PinDirectionSet()	Sets the direction (input/output) of a pin.
R_GPIO_PinControl()	Changes various settings for a pin (e.g. internal pull-up, open drain).
R_GPIO_GetVersion()	Returns the current version of this module.

3.2 R_GPIO_PortWrite

This function writes the levels of all pins on a port.

Format

```
void   R_GPIO_PortWrite(gpio_port_t port, uint8_t value);
```

Parameters

port

Which port to write to. See Section 2.10.1.

value

The value to write to the port. Each bit corresponds to a pin on the port (e.g. bit 0 of value will be written to pin 0 on supplied port)

Return Values

None.

Properties

Prototyped in file "r_gpio_rx_if.h"

Description

The input value will be written to the specified port. Each bit in the value parameter corresponds to a pin on the port. For example, bit 7 of write value corresponds to pin 7, bit 6 corresponds to pin 6, and so forth.

Reentrant

Function is re-entrant for different ports.

Example

```
/* Write 0xAA to Port 5. */
R_GPIO_PortWrite(GPIO_PORT_5, 0xAA);
```

Special Notes:

In the interest of performance, this function does not automatically check for non-existent pins when the port-wide write function is called. It is up to the user's application to insure that only valid pins are written to.

3.3 R_GPIO_PortRead

This function reads the levels of all pins on a port.

Format

```
uint8_t   R_GPIO_PortRead(gpio_port_t port);
```

Parameters

port

Which port to read. See Section 2.10.1.

Return Values

The value of the port.

Properties

Prototyped in file “r_gpio_rx_if.h”

Description

The specified port will be read, and the levels for all the pins will be returned. Each bit in the returned value corresponds to a pin on the port. For example, bit 7 of read value corresponds to pin 7, bit 6 corresponds to pin 6, and so forth.

Reentrant

Function is re-entrant for different ports.

Example

```
uint8_t    port_5_value;

/* Read Port 5. */
port_5_value = R_GPIO_PortRead(GPIO_PORT_5);
```

Special Notes:

None

3.4 R_GPIO_PortDirectionSet

This function sets multiple pins on a port to inputs or outputs at once.

Format

```
void   R_GPIO_PortDirectionSet(gpio_port_t port, gpio_dir_t dir, uint8_t mask);
```

Parameters

port

Which port to use. See Section 2.10.1.

dir

Which direction to use. See Section 2.10.5.

mask

Mask of which pins to change. 1 = set direction, 0 = do not change.

Return Values

None.

Properties

Prototyped in file "r_gpio_rx_if.h"

Description

Multiple pins on a port can be set to inputs or outputs at once. Each bit in the mask parameter corresponds to a pin on the port. For example, bit 7 of mask corresponds to pin 7, bit 6 corresponds to pin 6, and so forth. If a bit is set to 1 then the corresponding pin will be changed to an input or output as specified by the dir parameter. If a bit is set to 0 then the direction of the pin will not be changed.

Reentrant

Function is re-entrant for different ports.

Example

```
/* Set Pins 0, 1, and 5 as inputs on Port A. */
R_GPIO_PortDirectionSet(GPIO_PORT_A, GPIO_DIRECTION_INPUT, 0x23);

/* Set Pins 2, 3, 4, 6, and 7 as outputs on Port A. */
R_GPIO_PortDirectionSet(GPIO_PORT_A, GPIO_DIRECTION_OUTPUT, 0xDC);
```

Special Notes:

This function does not allow the user to specify the use of special modes such as input pull-up resistors or open-drain outputs. To enable these modes use the R_GPIO_PinControl() function.

3.5 R_GPIO_PinWrite

This function sets the level of a pin.

Format

```
void   R_GPIO_PinWrite(gpio_port_pin_t pin, gpio_level_t level);
```

Parameters

pin

Which pin to use. See Section 2.10.2.

level

What level to set the pin to.

Return Values

None.

Properties

Prototyped in file “r_gpio_rx_if.h”

Description

Pins can either be set as high (‘1’) or low (‘0’).

Reentrant

Function is re-entrant for different pins.

Example

```
/* Set Port E Pin 0 high. */
R_GPIO_PinWrite(GPIO_PORT_E_PIN_0, GPIO_LEVEL_HIGH);

/* Set Port 3 Pin 2 low. */
R_GPIO_PinWrite(GPIO_PORT_3_PIN_2, GPIO_LEVEL_LOW);
```

Special Notes:

None.

3.6 R_GPIO_PinRead

This function reads the level of a pin.

Format

```
gpio_level_t   R_GPIO_PinRead(gpio_port_pin_t pin);
```

Parameters

pin

Which pin to use. See Section 2.10.2.

Return Values

The level of the specified pin.

Properties

Prototyped in file “r_gpio_rx_if.h”

Description

The specified pin will be read and the level returned.

Reentrant

Function is re-entrant for different pins.

Example

```
/* Check level of Port 5 Pin 4. */
if (R_GPIO_PinRead(GPIO_PORT_5_PIN_4) == GPIO_LEVEL_HIGH)
{
    …
}
else
{
    …
}
```

Special Notes:

None.

3.7 R_GPIO_PinDirectionSet

This function sets the direction (input/output) of a pin.

Format

```
void   R_GPIO_PinDirectionSet(gpio_port_pin_t pin, gpio_dir_t dir);
```

Parameters

pin

Which pin to use. See Section 2.10.2.

dir

Which direction to use for this pin. See Section 2.10.5.

Return Values

None.

Properties

Prototyped in file "r_gpio_rx_if.h"

Description

This function sets pins as inputs or outputs. For enabling other settings such as open-drain outputs or internal pull-ups see the R_GPIO_PinControl() function.

Reentrant

Function is re-entrant for different pins.

Example

```
/* Set Port E Pin 0 as an output. */
R_GPIO_PinDirectionSet(GPIO_PORT_E_PIN_0, GPIO_DIRECTION_OUTPUT);

/* Set Port 3 Pin 2 as an input. */
R_GPIO_PinDirectionSet(GPIO_PORT_3_PIN_2, GPIO_DIRECTION_INPUT);
```

Special Notes:

None

3.8 R_GPIO_PinControl

This function allows the user to control various settings of a pin.

Format

```
gpio_err_t   R_GPIO_PinControl(gpio_port_pin_t pin, gpio_cmd_t cmd);
```

Parameters

pin Which pin to use. See Section 2.10.2.

cmd Which command to execute for this pin. See Section 2.10.6 for available commands.

Return Values

GPIO_SUCCESS:	*Successful; pin modified as specified by command.*
GPIO_ERR_INVALID_MODE:	*Error; this pin does not support the specified option.*
GPIO_ERR_INVALID_CMD:	*Error; the input command is not supported.*

Properties

Prototyped in file "r_gpio_rx_if.h"

Description

Depending on the MCU, pins have various settings that can be configured other than the direction and output level. Some examples include enabling open-drain outputs, internal pull-ups, and changing drive capacity levels. These features vary per chip which means that the options for this function will also vary.

Reentrant

Function is re-entrant for different pins.

Example

```
gpio_err_t gpio_err;

/* Set Port E Pin 0 as a CMOS output (default). */
R_GPIO_PinDirectionSet(GPIO_PORT_E_PIN_0, GPIO_DIRECTION_OUTPUT);
gpio_err |= R_GPIO_PinControl(GPIO_PORT_E_PIN_0, GPIO_CMD_OUT_CMOS);

/* Configure Port E Pin 0 as a high-current output */
gpio_err |= R_GPIO_PinControl(GPIO_PORT_E_PIN_0, GPIO_CMD_DSCR_ENABLE);

/* Configure Port E Pin 0 as a high-speed interface high-drive output */
gpio_err |= R_GPIO_PinControl(GPIO_PORT_E_PIN_0, GPIO_CMD_DSCR2_ENABLE);

/* Set Port E Pin 1 as a P-channel open-drain output. */
R_GPIO_PinDirectionSet(GPIO_PORT_E_PIN_1, GPIO_DIRECTION_OUTPUT);
gpio_err |= R_GPIO_PinControl(GPIO_PORT_E_PIN_1,GPIO_CMD_OUT_OPEN_DRAIN_P_CHAN);

/* Set Port E Pin 2 as an N-channel open-drain output. */
R_GPIO_PinDirectionSet(GPIO_PORT_E_PIN_2, GPIO_DIRECTION_OUTPUT);
gpio_err |= R_GPIO_PinControl(GPIO_PORT_E_PIN_2,GPIO_CMD_OUT_OPEN_DRAIN_N_CHAN);

/* Set Port 3 Pin 2 as input with pull-up disabled (default). */
R_GPIO_PinDirectionSet(GPIO_PORT_3_PIN_2, GPIO_DIRECTION_INPUT);
gpio_err |= R_GPIO_PinControl(GPIO_PORT_3_PIN_2, GPIO_CMD_IN_PULL_UP_DISABLE);

/* Set Port 3 Pin 3 as input with pull-up enabled. */
R_GPIO_PinDirectionSet(GPIO_PORT_3_PIN_3, GPIO_DIRECTION_INPUT);
gpio_err |= R_GPIO_PinControl(GPIO_PORT_3_PIN_3, GPIO_CMD_IN_PULL_UP_ENABLE);

/* Port 2 Pin 6 will be used as TXD1 for SCI peripheral. */
R_GPIO_PinDirectionSet(GPIO_PORT_2_PIN_6, GPIO_DIRECTION_OUTPUT);
gpio_err |= R_GPIO_PinControl(GPIO_PORT_2_PIN_6, GPIO_CMD_ASSIGN_TO_PERIPHERAL);

/* Port 5 Pin 4 will be used as GPIO. */
gpio_err |= R_GPIO_PinControl(GPIO_PORT_5_PIN_4, GPIO_CMD_ASSIGN_TO_GPIO);

/* GPIO_SUCCESS is set to 0 so if gpio_err is not 0 then an error occurred above.
You could check gpio_err after every function call if needed. */
if (GPIO_SUCCESS != gpio_err)
{   /* Handle the error.  */}
```

3.9 R_GPIO_GetVersion

Returns the current version of this API.

Format

```
uint32_t   R_GPIO_GetVersion(void);
```

Parameters

None.

Return Values

Version of this API.

Properties

Prototyped in file “r_gpio_rx_if.h”

Description

This function will return the version of the currently running API. 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

Yes.

Example

```
uint32_t cur_version;

/* Get version of running r_gpio_rx API. */
cur_version = R_GPIO_GetVersion();

/* Check to make sure version is new enough for this application’s use. */
if (MIN_VERSION > cur_version)
{
    /* This r_gpio_rx version is not new enough and does not have XXX feature
       that is needed by this application. Alert user. */
    ….
}
```

Special Notes:

This function is specified to be an inline function in r_gpio_rx.c.

4. Demo Projects

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

4.1 gpio_demo_rskrx113

The **gpio_demo_rskrx113** program demonstrates how to set the direction of an IO port as an input or output and how to read or write it. Once the demo code has been compiled and down-loaded to the target board and is running, the demo will flash LED2 three times to show that the demo is running then, wait for key-presses on SW1. LED2 is turned on while SW1 is pressed and off when it is released.

4.2 gpio_demo_rskrx231, gpio_demo_rskrx64m, gpio_demo_rskrx71m

The **gpio_demo_rskrx231, gpio_demo_rskrx64m** and **gpio_demo_rskrx71m** demo programs operate the same. They demonstrate how to set up a port pin as an input or output and how to read or write it. They also demonstrate how to configure an output pin for high-current drive.
Once the code has been compiled and down-loaded to the target board and is running, LED3 will flash three times to show that the demo is running then, wait for key-presses on SW1. LED3 is turned on while SW1 is pressed and off when it is released.

4.3 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”.

5. Provided Modules

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

6. Reference Documents

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

Related Technical Updates

This module reflects the content of the following technical updates.

None

Website and Support

Renesas Electronics Website

http://www.renesas.com/

Inquiries

http://www.renesas.com/contact

All trademarks and registered trademarks are the property of their respective owners.

Revision Record

Rev.	Date	Description Page	 Summary
1.00	Nov 15, 2013	—	First Release
1.20	April 23 ,2014	1	Additional supported MCUs listed. List of documents revised.
		3	Updated list of required BSP versions.
		3	Added note on limitation: Port Switching feature not supported.
		4-7	Corrected value assigned in references to PORT_J
		7	Added section 2.9.3 Port_Pin Masks
		11	Added note regarding writing to port bits for non-existent pins
1.30	June 3, 2014	1	Added mention of RX64M in the list of supported MCUs.
		3	Updated toolchain version.
		19	Updated formatting of section 3.9.
1.40	Nov 28, 2014	—	Added support for the RX113 Group.
		5	Added a Code Size section.
1.50	Mar 06, 2015	—	Added support for the RX71M Group.
		5	Updated the Code Size table for RX71M.
		10	Updated “Control Commands” to include DSCR.
1.60	June 30, 2015	—	Added support for the RX231 Group.
		5	Updated the Code Size table for RX231.
1.70	Sep 30, 2015	—	Added support for the RX23T Group.
		5	Updated the Code Size table for RX23T.
1.80	Oct 1, 2015	—	Added support for the RX130 Group.
		5	Updated the Code Size table for RX130.
1.90	Dec 1, 2015	—	Added support for the RX24T Group.
		1, 10	Changed the document number for the “Board Support Package Firmware Integration Technology Module” application note.
		4	Changed the description in section 2.
		5	Updated the Code Size table for RX24T.
		20	Added “4. Demo Projects”.
2.00	Feb 1, 2016	—	Added support for the RX230 Group.
		5	Updated the Code Size table for RX230.
		21	Added “Related Technical Updates”.
2.01	June 15, 2016	20	Added RSKRX64M to “4. Demo Projects”.
2.10	Oct 1, 2016	—	Added support for the RX65N Group.
		1	Changed the document number for the “Board Support Package Firmware Integration Technology Module” application note. Added RX65N Group to the target device.
		4	Added RX65N to “2.5 Supported Toolchains”.
		5	Updated the Code Size table for RX65N.
		10	Updated “Control Commands” to include DSCR2.
		19	Added DSCR2 command example to “3.8 R_GPIO_PinControl”.
		22	Added “5. Provided Modules”. Added “6. Reference Documents”.
		23	Updated Inquiries.

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

Notice

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

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

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

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

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

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

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

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

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

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

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

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

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

© 2016 Renesas Electronics Corporation. All rights reserved.

Colophon 5.0