FIT Sample Code

r\_switches

Overview

This code allows you to easily have callback functions for when a switch is pressed on your board. The RX has IRQ pins that will trigger an interrupt when the voltage on the pin changes. The pin can be setup to trigger on rising edges, falling edges, rising and falling edges, or low level. This code sets all of this up for the user. The user only needs to specify what function should be called when a switch is pressed.

Supported Boards

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

* **RSKRX610**
* **RSK+RX62N**
* **RSKRX62T**
* **RDKRX62N**
* **RSKRX630**
* **RSKRX63N**
* **RDKRX63N**

Contents

[1. API Information 2](#_Toc319498403)

[2. API Functions 3](#_Toc319498404)

[Website and Support 4](#_Toc319498405)

# API Information

This Middleware API follows the Renesas API naming standards.

Hardware Requirements

This middleware requires your MCU support the following features:

* Trigger an interrupt based on voltage on pin or change of voltage on pin.

Hardware Resource Requirements

None.

Header Files

All API calls are accessed by including a single file *r\_switches\_if.h* which is supplied with this middleware’s project code.

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*.

Configuration Overview

All configuration is done through the header file *r\_switches\_config.h*.

|  |  |
| --- | --- |
| **Configuration Options in *r\_switches\_config.h*** | |
| **SW1\_CALLBACK\_FUNCTION**  **SW2\_CALLBACK\_FUNCTION**  **. . .**  **SWn\_CALLBACK\_FUNCTION** | The definition for these macros should be the name of the function that you want called when a switch is pressed. It is very important that the user recognize that this function will be called from the interrupt service routine. This means that code inside of the function should be kept short to ensure it does not hold up the rest of the system.  Example: If SW1\_CALLBACK\_FUNCTION is defined to be sw1\_callback then the sw1\_callback function will be called when switch 1 is pressed. |

Table : Configuration Options

API Data Structures

None.

Return Values

None.

Adding Middleware to Your Project

Follow the steps below to add the middleware’s code to your project.

1. Copy the ‘r\_switches’ directory to your project directory.
2. Add src\r\_switches.c to your project.
3. Add an include path to the 'r\_switches' directory.
4. Add an include path to the 'r\_switches\src' directory.
5. Configure middleware through r\_switches\_config.h.
6. Add a #include for r\_switches\_if.h to files that need to use this package.

# API Functions

R\_SWITCHES\_Init

Initializes port pins for switches and enables IRQ interrupts.

Format

void R\_SWITCHES\_Init(void);

Parameters

None.

Return Values

None.

Properties

Prototyped in file “r\_switches\_if.h”

Description

Initializes everything needed to enable interrupts when a switch is pressed on your board.

Reentrant

Yes.

Example

/\* Enable interrupts when switch is pressed. \*/

R\_SWITCHES\_Init();

R\_SWITCHES\_GetVersion

Returns the version of the code.

Format

uint32\_t R\_SWITCHES\_GetVersion(void);

Parameters

None.

Return Values

Four byte value that represents the version of the code.

Properties

Prototyped in file “r\_switches\_if.h”

Description

Returns an encoded version of the middleware. 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

/\* Get version of r\_switches code and make sure it’s greater than v2.08 \*/

uint32\_t current\_version;

current\_version = R\_SWITCHES\_GetVersion();

if (current\_version <= 0x00020008)

{

printf(“Need at least v2.08 of r\_switches code to run this application!”);

}

# Website and Support

Renesas Electronics Website

<http://www.renesas.com/>

Inquiries

<http://www.renesas.com/inquiry>

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

Revision Record

|  |  |  |  |
| --- | --- | --- | --- |
| Rev. | Date | Description | |
| Page | Summary |
| 1.00 | Mar.13.12 | — | First edition issued |
|  |  |  |  |
|  |  |  |  |

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

C:\Documents and Settings\b1900215.RKS-DOMAIN\Desktop\AN_templete\E_address.eps

1. Handling of Unused Pins

Handle unused pins in accord 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 one with a different type number, confirm that the change will not lead to problems.

* The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.