RZ/A2M Group

RZ/A2M INTC Driver

R01AN4500EG0101
Rev.1.01
June 10, 2020

Introduction

This application note describes the operation of the software INTC Driver for the RZ/A2 device on the RZ/A2M CPU Board.

It provides a comprehensive overview of the driver. For further details please refer to the software driver itself.

The user is assumed to have knowledge of e^2 studio and to be equipped with an RZ/A2M CPU Board.

Target Device

RZ/A2M Group

Driver Dependencies

This driver has no other driver dependencies.

Referenced Documents

Document Type	Document Name	Document No.
User's Manual	RZ/A2M Hardware Manual	R01UH0746EJ

List of Abbreviations and Acronyms

Abbreviation	Full Form
API	Application Programming Interface
ARM	Advanced RISC Machines
CPU	Central Processing Unit
HLD	High Layer Driver
IDE	Integrated Development Environment
INTC	Interrupt Controller
LLD	Low Layer Driver

Table 1-1 List of Abbreviations and Acronyms

Contents

1. Outline of Software Driver

The INTC (Interrupt Controller) driver is an abstraction layer between the application and the hardware. It provides functions for enabling and disabling interrupts, setting callback functions, interrupt priorities, etc.

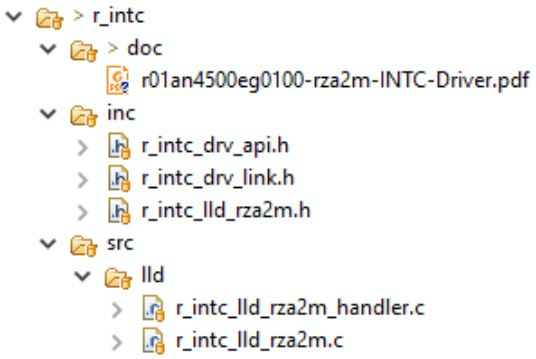
2. Description of the Software Driver

The key features of the driver include:

- Enabling and disabling interrupts
- Setting interrupt detection methods (such as level or edge triggered)
- Setting interrupt priorities
- Setting interrupt callback functions
- Masking interrupts on or off

2.1 Structure

Unlike many of the other drivers, the INTC driver currently consists of a single layer: the Low Layer Driver (LLD). This includes all the hardware specific functions and provides the API to the application.



2.2 Description of each file

Each file's description can be seen in the following table.

Filename	Usage	Description
Application-Facing Driver API		
r_intc_drv_api.h	Application	** FOR FUTURE USE **
Low Layer API		
r_intc_lld_xxxx.h	API header file	Low Layer Driver (LLD) header file (where "xxxx" is a device and board-specific identification). This is the header file to include in application code.
Abstraction Link between High and Low Layer Drivers (HLD/LLD Link)		
r_intc_drv_link.h	Private (HLD/LLD only)	** FOR FUTURE USE **
Low Layer Driver (LLD) Source		
r_intc_lld_xxxx.c	Private (LLD only)	(Where "xxxx" is a device and board specific identification). Provides the definitions for the Low Layer Driver interface.
r_intc_lld_xxxx_handler.c	Private (LLD only)	Contains Low Layer Driver interrupt handler routines

2.3 Low Layer Driver

The Low Layer Driver provides the functions to configure the hardware.

Return Type	Function	Description	Arguments	Return
e_r_drv_intc_err_t	**R_INTC_Init(**void**)**	Initialise the INTC driver	None	**INTC_SUCCESS**
e_r_drv_intc_err_t	**R_INTC_SetNMIConfig(**const st_r_drv_nmi_cfg_t * p_nmi_config**)**	Set NMI configuration	**p_nmi_config:** [in] NMI configuration (rising or falling edge)	**INTC_SUCCESS**
e_r_drv_intc_err_t	**R_INTC_GetNMIConfig(**st_r_drv_nmi_cfg_t * p_nmi_config**)**	Set NMI configuration	**p_nmi_config:** [out] NMI configuration (rising or falling edge)	**INTC_SUCCESS**
e_r_drv_intc_err_t	**R_IRQ_Init(**const st_r_drv_irq_cfg_t * p_irq_config**)**	Initialise IRQs (IRQ0 – IRQ7) setting interrupt detection methods	**p_irq_config:** [in] Interrupt detection methods	**INTC_SUCCESS**
e_r_drv_intc_err_t	**R_TINT_Init(**const st_r_drv_tint_cfg_t * p_tint_config**)**	Initialise TINTs (TINT0 – TINT31) setting interrupt detection methods	**p_tint_config:** [in] Interrupt detection methods	**INTC_SUCCESS**
e_r_drv_intc_err_t	**R_INTC_RegistIntFunc(**e_r_drv_intc_intid_t int_id, void (* func) (uint32_t int_sense))	Register interrupt handler function	**int_id:** [in] interrupt ID (0 – 511) **int_sense:** [in] interrupt handler function	**INTC_SUCCESS or INTC_ERR_INVALID**
e_r_drv_intc_err_t	**R_INTC_GetIntFunc(**e_r_drv_intc_intid_t int_id, void (* func) (uint32_t int_sense)**)**	Get the address of the interrupt handler function	**int_id:** [in] interrupt ID (0 – 511) **int_sense:** [out] interrupt handler function	**INTC_SUCCESS, INTC_ERR_INVALID or INTC_ERR_UNASSIGNED_CALLBACK**
e_r_drv_intc_err_t	**R_INTC_Enable(**e_r_drv_intc_intid_t int_id**)**	Enable an interrupt	**int_id:** [in] interrupt ID (0 – 511)	**INTC_SUCCESS or INTC_ERR_INVALID**
e_r_drv_intc_err_t	**R_INTC_Disable(**e_r_drv_intc_intid_t int_id**)**	Disable an interrupt	**int_id:** [in] interrupt ID (0 – 511)	**INTC_SUCCESS or INTC_ERR_INVALID**
e_r_drv_intc_err_t	**R_INTC_SetPriority(**e_r_drv_intc_intid_t int_id, e_r_drv_intc_priority_t priority**)**	Set an interrupt priority	**int_id:** [in] interrupt ID (0 – 511) **priority:** [in] interrupt priority	**INTC_SUCCESS, INTC_ERR_INVALID or INTC_ERR_INVALID_PRIORITY**
e_r_drv_intc_err_t	**R_INTC_GetPriority(**e_r_drv_intc_intid_t int_id, e_r_drv_intc_priority_t *priority**)**	Get an interrupt priority	**int_id:** [in] interrupt ID (0 – 511) **priority:** [out] interrupt priority	**INTC_SUCCESS, INTC_ERR_INVALID, INTC_ERR_INVALID_PRIORITY or INTC_ERR_INVALID**

e_r_drv_intc_err_t	**R_INTC_SetMaskLevel(**e_r_drv_intc_priority_t mask_level**)**	Sets the interrupt mask level	**mask_level:** [in] Interrupt mask level (0 - 31)	**INTC_SUCCESS or INTC_ERR_INVALID_PRIORITY**
e_r_drv_intc_err_t	**R_INTC_GetMaskLevel(**e_r_drv_intc_priority_t *mask_level**)**	Gets the interrupt mask level	**mask_level:** [out] Interrupt mask level (0 - 31)	**INTC_SUCCESS**
e_r_drv_intc_err_t	**R_IRQ_SetSense(**e_r_drv_irq_num_t irq_num, e_r_drv_irq_sense_t sense**)**	Set IRQ pin interrupt detection method	**num:** [in] IRQ pin number **sense:** [in] interrupt detection method	**INTC_SUCCESS, INTC_ERR_INVALID_NUM or INTC_ERR_INVALID_SENSE**
e_r_drv_intc_err_t	**R_IRQ_GetSense(**e_r_drv_irq_num_t irq_num, e_r_drv_irq_sense_t *sense**)**	Get IRQ pin interrupt detection method	**num:** [in] IRQ pin number **sense:** [out] interrupt detection method	**INTC_SUCCESS or INTC_ERR_INVALID_NUM**
e_r_drv_intc_err_t	**R_INTC_GetPendingStatus(**e_r_drv_intc_intid_t int_id, e_r_drv_intc_pending_t *pending_status**)**	Get interrupt pending state	**int_id:** [in] interrupt ID (0 – 511) **pending_status:** [out] the pending state	**INTC_SUCCESS or INTC_ERR_INVALID_ID**
e_r_drv_intc_err_t	**R_TINT_SetSense(**e_r_drv_tint_num_t tint_num, e_r_drv_tint_sense_t sense**)**	Set the pin interrupt detection method	**tint_num:** [in] TINT pin number **sense:** [in] TINT interrupt detection method	**INTC_SUCCESS, INTC_ERR_INVALID_NUM or INTC_ERR_INVALID_SENSE**
e_r_drv_intc_err_t	**R_INTC_SetIrqMask(**e_r_drv_irq_mask_t mask**)**	Set IRQ mask	**mask:** [in] interrupt mask (on or off)	**INTC_SUCCESS or INTC_ERR_INVALID**
e_r_drv_intc_err_t	**R_INTC_GetIrqMask(**e_r_drv_irq_mask_t *mask**)**	Get IRQ mask	**mask:** [in] interrupt mask (on or off)	**INTC_SUCCESS**
uint32_t	**R_INTC_GetVersion(**st_drv_info_t *pinfo**)**	Get Low Layer Driver version information	**pinfo**: [out] pointer to version information structure	**DRV_SUCCESS**

The possible return values used in this driver are listed below:

Return Code	Description
DRV_SUCCESS, INTC_SUCCESS	**Function execution successful.**
INTC_ERR_INVALID	**At least one argument invalid.**
INTC_ERR_INVALID_PRIORITY	**Invalid interrupt priority level specified by argument.**
INTC_ERR_INVALID_ID	**The interrupt ID specified by argument is invalid for the device.**
INTC_ERR_INVALID_NUM	**Invalid TINT pin group number/IRQ pin number specified by argument.**
INTC_ERR_INVALID_SENSE	**Invalid detection method specified by argument.**
INTC_ERR_UNASSIGNED_CALLBACK	**No callback function assigned to the interrupt ID specified by argument.**

3. Example of Use

This section gives simple examples for initialising the driver, registering an interrupt callback function, setting interrupt priority, enabling an interrupt, disabling an interrupt, and finally getting the driver version.

3.1 Initialise Driver

```
e_r_drv_intc_err_t result;

result = R_INTC_Init();
```

3.2 Registering an Interrupt Callback Function

```
void MyInterruptHandler(uint32_t int_sense)
{
    /* interrupt handling code */
}

result = R_INTC_RegistIntFunc(INTC_ID_DMAC30_DMAERR0, MyInterruptHandler);
```

3.3 Setting Interrupt Priority

```
result = R_INTC_SetPriority(INTC_ID_DMAC30_DMAERR0, 28);
```

3.4 Enabling an Interrupt

```
result = R_INTC_Enable(INTC_ID_DMAC31_DMAINT9);
```

3.5 Disabling an Interrupt

```
result = R_INTC_Disable(INTC_ID_DMAC31_DMAINT9);
```

3.6 Get Version

```
st_ver_info_t info;
uint32_t get_version_result;

get_version_result = R_INTC_GetVersion(&Info);
```

4. OS Support

This driver supports any OS.

5. How to Import the Driver

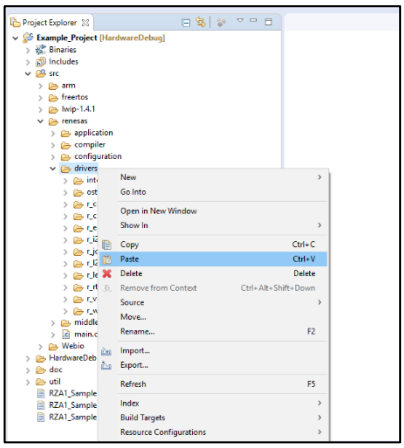
This section describes how to import the driver into your project. Generally, there are two steps in any IDE:

1) Copy the software driver to the location in the source tree that you require for your project.
2) Add the include path of the driver to the compiler.

5.1 e² studio

To import the driver into your project please follow the instructions below.

1) In Windows Explorer, right-click on the r_intc folder, and click **Copy**.
2) In e² studio Project Explorer view, select the folder where you wish the driver project to be located; right-click and click **Paste**.
3) Right-click on the parent project folder (in this case 'Example_Project') and click **Properties ...**
4) In 'C/C++ Build → Settings → Cross ARM Compiler → Includes', add the include folder of the newly added driver, e.g. '${ProjDirPath}\src\renesas\drivers\r_intc\inc'



Website and Support

Renesas Electronics website
https://www.renesas.com/

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

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Sept 19, 2018	All	Created document.
1.01	June 10, 2020	Pages 6 and 7	Updated R_INTC_GetIntFunc return values, added return code table.

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

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

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

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

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

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

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

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

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

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

Trademarks

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

Contact information

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