RX Family

Using Register Bank Save Function

Introduction

Except in some products, the RXv3 CPU provides register bank save function in order to perform fast collective saving and restoring of CPU registers. This application note describes these functions and their use in interrupt handlers.

Unless otherwise indicated, this application note refers to the RX72T Group. For information regarding the specifications of other MCUs, see the hardware edition of the user's manual for those particular MCUs.

Target Device

RX Family with register bank save function

Contents

1. Register Bank Save Function

In order to perform fast collective saving and restoring of CPU registers, the RXv3 CPU provides dedicated save register banks and instructions for using these banks. Using these save register banks, it is possible to perform fast register saving at the beginning of interrupt handlers, and high-speed register restoring at the end of interrupt handlers.

1.1 Save Register Banks

The save register banks can only be accessed with the SAVE instruction and RSTR instruction. Each of these banks is used to save and restore the values of the following CPU registers: all general purpose registers (R1 to R15) except for R0, the USP, the FPSW, and the accumulators (ACC0, ACC1).

Each save register bank is assigned a unique number (bank number). For the RX72T Group, 16 save register banks are provided, as shown in Figure 1.1.

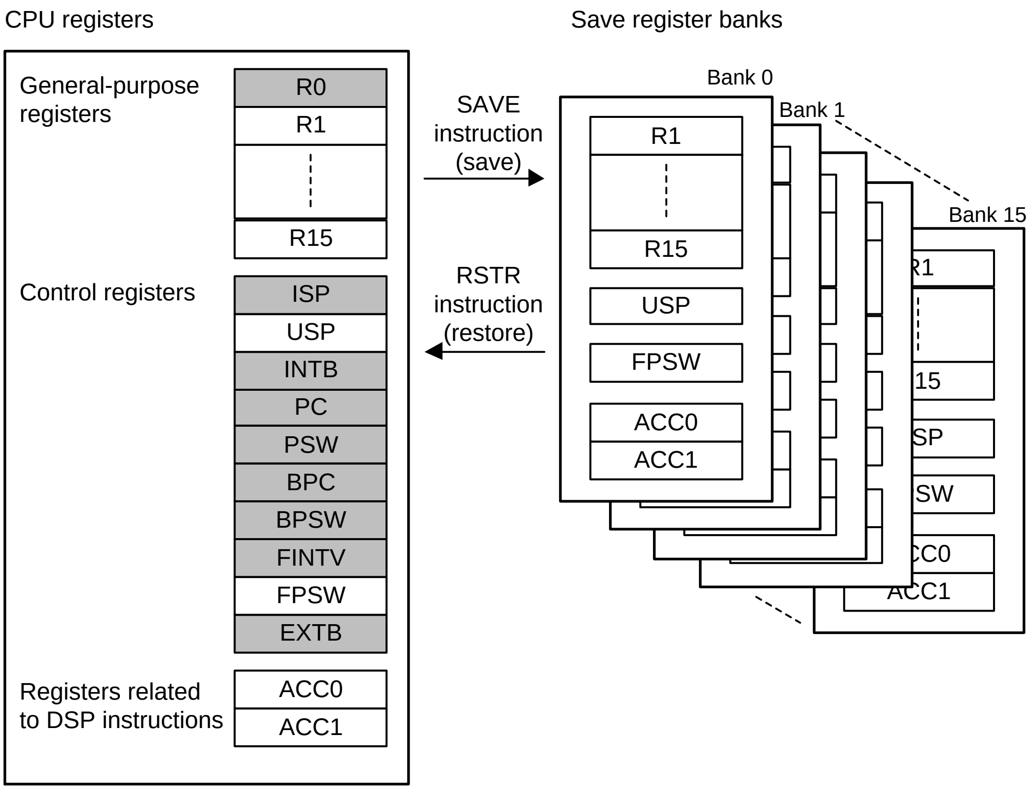

Figure 1.1 RX72T Group Save Register Banks

1.2 Saving and Restoring Registers

The RXv3 CPU is equipped with one buffer for register saving. If the SAVE instruction has been executed, the registers are saved to the specified save register bank via this buffer. Figure 1.2 shows an example of saving and restoring registers via the buffer. If the bank number that was last specified by the SAVE instruction is specified by the RSTR instruction, the registers can be restored quickly by restoring them not from the save register bank but rather from the buffer. If, on the other hand, a bank number other than the last one that was specified by the SAVE instruction is specified by the RSTR instruction, the registers will be restored not from the buffer but rather from the save register bank.

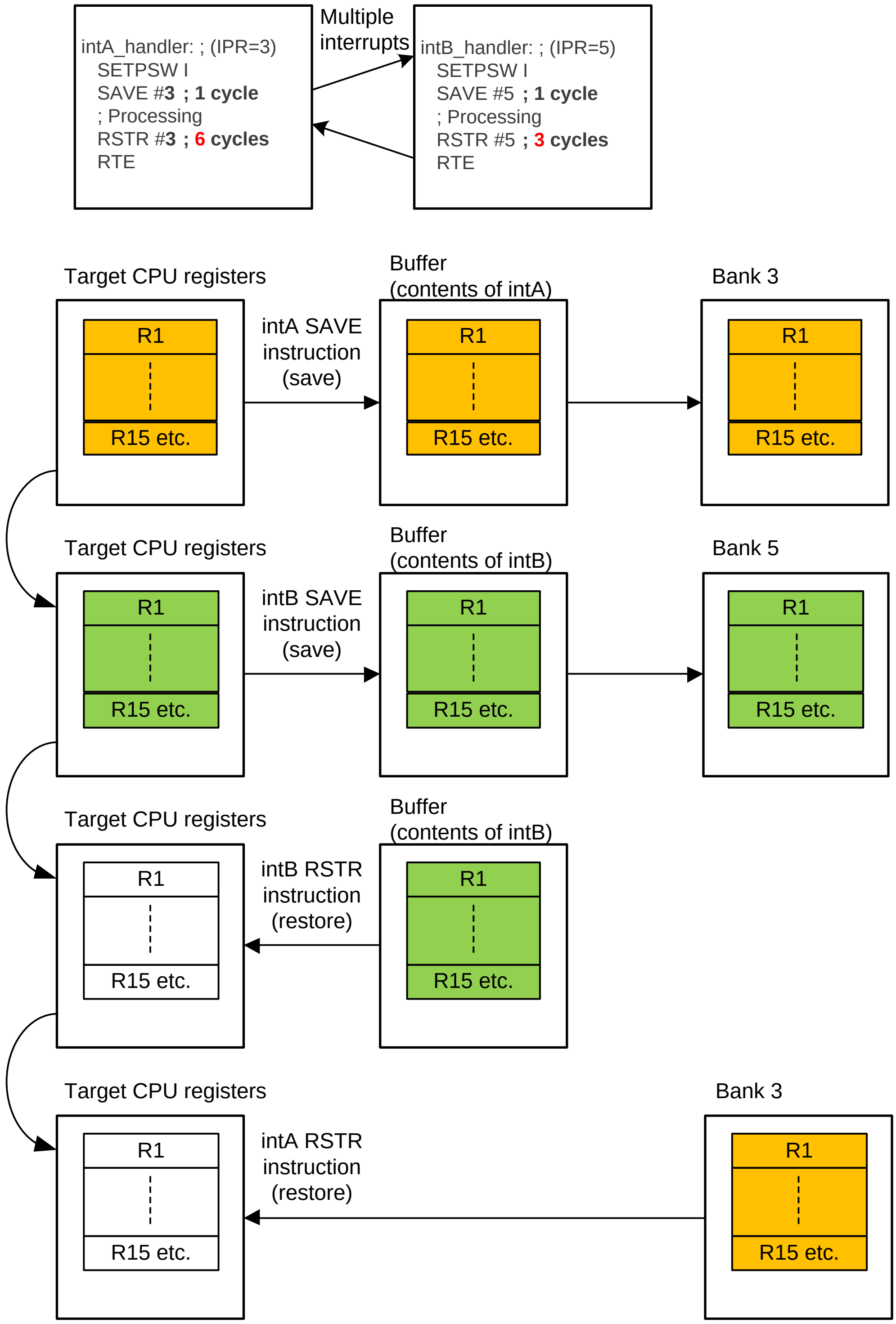

Figure 1.2 Example of Saving and Restoring Registers via Buffer

2. Using Save Register Banks in Interrupt Handlers

2.1 Interrupt Handlers

Figure 2.1 shows an overview of an interrupt handler. Collective saving and restoring of registers is performed using a SAVE instruction and RSTR instruction at the beginning and end of an interrupt handler. For details of SAVE and RSTR instructions, see RX Family RXv3 Instruction Set Architecture User's Manual: Software. Within any particular interrupt handler, be sure to specify the same bank number in the SAVE instruction and RSTR instruction.

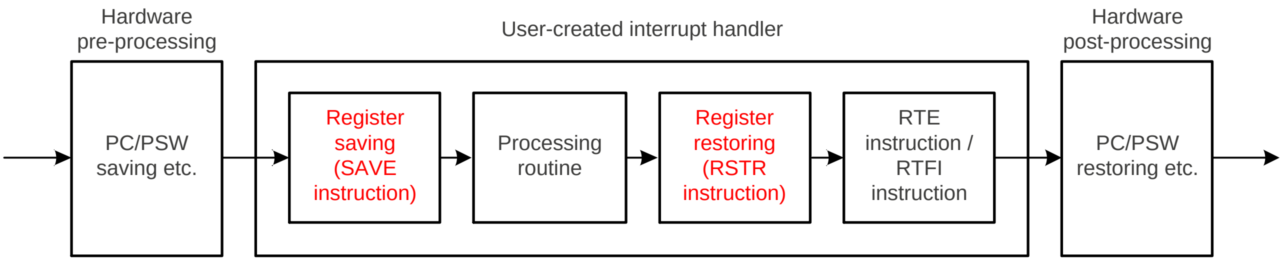

Figure 2.1 Overview of Interrupt Handler

2.2 Assigning Bank Numbers

If a save register bank is used in an interrupt handler, before restoring the saved registers it is necessary to ensure that the bank is not erroneously overwritten when multiple interrupts occur.

As shown in Table 2.1, if bank numbers are assigned such that the interrupt priority level of the interrupt handler matches the bank number used in that interrupt handler, the save register banks can be used by all of the interrupt handlers. Moreover, since multiple interrupts of the same interrupt priority level will not occur, there is no risk that the registers that have been saved will be overwritten.

Table 2.1 Example of Recommended Bank Number Assignment Method

Interrupt Priority Level of Interrupt Handler	Bank Number Used by Interrupt Handler
1	1
2	2
3	3
:	:
15	15

For example, if there are three interrupt handlers for interrupt A (priority level 5), interrupt B (priority level 6), and interrupt C (priority level 5), respectively, interrupts A and C will use bank 5, and interrupt B will use bank 6.

2.3 Execution Cycle-Reducing Effect of Using Save Register Banks

When save register banks are not used, the number of execution cycles required to save and restore registers depends upon the number of registers being saved. For this reason, the larger the number of registers that must be saved by an interrupt handler, the greater the effect of using a save register bank.

2.3.1 Execution Cycles Required to Save and Restore Registers

The number of execution cycles required to save and restore registers when save register banks are and are not used is shown below.

1. When save register banks are not used: 2N + 12A + 4C (cycles)
   Here, N is the number of target general purpose registers (R1 to R15), A is the number of target accumulators (ACC0, ACC1), and C is USP/FPSW target count.
   Note that the cycle count may increase depending on the usage conditions (for example, the stack placement address).

2. When save register banks are used: 4 to 7 cycles
   The cycle count will be 4 cycles if during execution of an interrupt handler another interrupt that uses the save register bank is not accepted, or 7 cycles otherwise.

2.3.2 Execution Cycle Count Comparative Example

An execution cycle count example for the case in which the 10 registers R1 to R10 are saved and restored is shown below. When a save register bank is used, saving and restoring is fast, executing in 4 cycles. It was thus possible in this example to reduce the cycle count by 16 cycles.

1. When save register banks are not used

```
interrupt_handler:
PUSHM      R1-R10        ; 10 cycles
    :                    ; Processing that uses R1 to R10
POPM       R1-R10        ; 10 cycles
RTE
```

2. When save register banks are used

```
interrupt_handler:
SAVE #1    ; 1 cycle
    :      ; Processing that uses R1 to R10
RSTR #1    ; 3 to 6 cycles
RTE
```

2.3.3 Determining Whether to Use Save Register Banks

It is recommended that save register banks be used when writing code in C (with the CC-RX compiler) without paying attention to the execution cycle count, or when any of the conditions below applies. In cases other than these, refer to 2.3.1 Execution Cycles Required to Save and Restore Registers to decide whether or not to use the save register banks.

If other interrupts (multiple interrupts) that use save register banks are used:

- If four or more general purpose registers are targeted for saving and restoring
- If one or more accumulators are targeted for saving and restoring
- If the USP and FPSW are among the targets for saving and restoring

If other interrupts (multiple interrupts) that use save register banks are not used:

- If two or more general purpose registers are targeted for saving and restoring
- If one or more accumulators are targeted for saving and restoring
- If the USP or FPSW is among the targets for saving and restoring

2.4 Execution Cycle Count When Multiple Interrupts Occur

Figure 2.2 shows the execution cycle counts of the SAVE and RSTR instructions if multiple interrupts occur in which some of the interrupts use the save register banks and some do not.

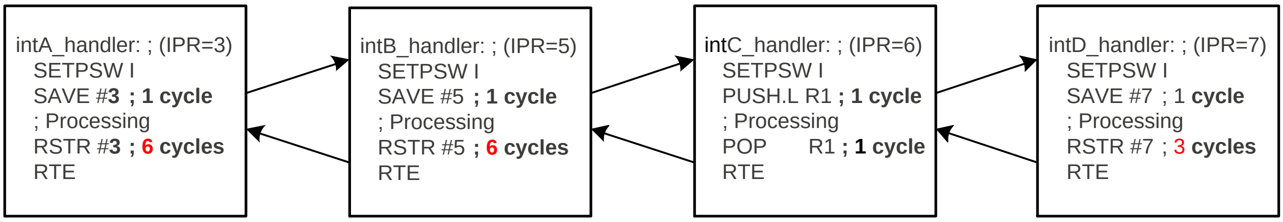

Figure 2.2 Execution Cycle Count When Multiple Interrupts Occur

If the bank number specified by the RSTR instruction is the same as the bank number specified by the most-recently executed SAVE instruction then the execution cycle count for the RSTR instruction will be 3 cycles, and if the bank number is different then the count will be 6 cycles.

3. Writing Code in C (CC-RX Compiler)

When “bank=N” (N = 0 to 15) is specified in “#pragma interrupt”, instructions that use the save register bank will be generated in the interrupt handler. For details, see the CC-RX Compiler User’s Manual.

Code example specifying vector number 64 and bank 3

```
#pragma interrupt handler(vect=64, bank=3)
    void handler(void) {

        :
    }
```

Compilation →

```
_handler:
 .RVECTOR      64, _handler
 SAVE   #03H
    :
 RSTR   #03H
 RTE
```

In an interrupt handler in which “bank=N” has been specified, the save register bank will be used regardless of the number of registers to be saved. For N, be sure to specify the bank number to be used. Moreover, only use “bank=N” with MCUs that provide register bank save function.

4. Important Notes

4.1 “-bank” Assembler Option

When writing code that utilizes the register bank save function without using an integrated development environment like CS+ or e^2 studio, specify the “-bank” assembler option. For details, see the CC-RX Compiler User’s Manual.

If an integrated development environment like CS+ or e^2 studio is used, the “-bank” assembler option will be added automatically.

4.2 Save Register Banks After Reset

The values in the save register banks after a reset are undefined. If a RSTR instruction is executed without executing a SAVE instruction, undefined values will be stored in registers R1 to R15 etc.

5. Reference Documents

User's Manual: Hardware
RX72T Group User's Manual: Hardware (R01UH0803)
(The latest version can be downloaded from the Renesas Electronics website.)

User's Manual: Software
RX Family RXv3 Instruction Set Architecture User's Manual: Software (R01US0316)
(The latest version can be downloaded from the Renesas Electronics website.)

User's Manual: C compiler
CC-RX Compiler User's Manual (R20UT3248)
(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.)

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Feb.28.19	—	First edition issued

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.

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/.