Application Note

Document No.: AN1133

G32R501 Interrupt Application Note

Version: V1.0

1 Introduction

This application note introduces G32R5xx interrupts and application design.

Contents

1 Introduction 1

2 NVIC 3

2.1 Interrupt Vector Table and Interrupt Priority 3

2.2 Interrupt Service Function 4

3 EXTI 5

3.1 Hardware Interrupt Selection 5

3.2 Hardware Event Selection 5

3.3 Software Interrupt/Event Selection 5

4 Software Configuration Process 6

5 Revision 8

2 NVIC

NVIC (Nested Vectored Interrupt Controller) controls the interrupt related functions of the entire chip, and is integrated into the core. The core of G32R5xx is Arm® Cortex®-M52. By programming the NVIC register, the interrupts can be enabled or the interrupt priority can be configured.

G32R5xx has 227 external interrupts (excluding 11 system exceptions).

2.1 Interrupt Vector Table and Interrupt Priority

If the peripheral interrupt function is used when programming the G32R5xx chip, the interrupt vector table in ROM needs to be copied to RAM, and the interrupt vector table in RAM needs to be remapped to the core VTOR register so that users can register new interrupt response functions.

The following table is an excerpt from the interrupt vector table:

Table 1 Interrupt Vector Table (excerpt)

Exception type	Vector No.	Priority	Description
-	-	-	Reserved
Reset	-	-15	Reset
NMI	-	-14	Non-maskable interrupt
HardFault	-	-13	Various hardware faults
MemManage	-	-12	Memory fault
BusFault	-	-11	Bus fault
UsageFault	-	-10	Instruction execution fault
SecureFault	-	-9	Data security fault
SVCall	-	-5	System service called by general SWIU instruction
Debug Monitor	-	-4	Debugging function exception
PendSV	-	-2	Pending system service
SysTick	-	-1	System tick timer
…	…	…	…
DCCOMP interrupt	11	Can set	DCCOMP interrupt
…	…	…	…
CTI INT0	225	Can set	CTI interrupt 0
CTI INT1	226	Can set	CTI interrupt 1

Arm® Cortex®-M52 allows users to set priority for each exception/interrupt, so that they can manage the interrupt responses more flexibly.

The G32R5xx interrupts provide multiple different interrupt priority grouping configurations, each of which defines the allocation method of preempting priority and sub-priority.

2.2 Interrupt Service Function

The four states of interrupt: Active, Pending, Inactive, Active and pending.

Active:

- Being processed.
- Not processed. The response program is preempted by a higher-priority exception/interrupt response program.

Pending: An exception/interrupt has been generated but not yet activated.

Inactive: No exception/interrupt has been generated yet.

Active and pending:

One instance of an exception/interrupt is in an active state, and the second instance of an exception/interrupt is in a suspended state. Only asynchronous exceptions/interrupts can be in both active and suspended states simultaneously. Synchronous exceptions/interrupts are invalid, suspended, or active.

When an exception/interrupt is active, the program will respond to the interrupt service function, and the processor will find the corresponding interrupt service function based on the interrupt vector table and jump to this function and execute it.

In the interrupt service function, the program will execute specific code to handle related events. After completing handling, the program usually needs to clear the corresponding interrupt flag to notify the system that the interrupt has been processed and is ready to accept the next interrupt.

After the interrupt service function is executed, the system will restore the previously saved context and continue executing the interrupted program code.

3 EXTI

When using the EXTI interrupt line, the interrupt line shall be configured and enabled to generate interrupts. Set 2 trigger registers according to the required edge detection, and write "1" in the corresponding bit of the interrupt mask register to enable interrupt requests. If the interrupt is enabled, when the selected signal edge appears on the EXTI interrupt line, an interrupt request will be generated, and the corresponding interrupt flag will be set to 1. Writing “1” in the corresponding bit of the suspend register will clear the interrupt flag.

To generate an event, first the event line shall be configured and enabled. Set 2 trigger registers according to the required edge detection, and write "1" in the corresponding bit of the event mask register to enable event requests. When a selected signal edge appears on the event line, an event pulse will be generated, and the corresponding suspended bit will not be set to 1.

All EXTI lines can also generate software interrupt/event requests by writing "1" to the software interrupt/event register in software.

3.1 Hardware Interrupt Selection

To configure a line as an interrupt source, please follow the following steps:

1. Configure the mask bit of the interrupt line.
2. Configure the trigger selection bit for the interrupt line.
3. Configure the enable bit and mask bit used for controlling the NVIC IRQ channel mapped to the extended interrupt controller (EXTI), so that the requests in any interrupt line can be responded to correctly.

3.2 Hardware Event Selection

To configure a line as an event source, please follow the following steps:

1. Configure the mask bit of the event line.
2. Configure the trigger selection bit for the event line.

3.3 Software Interrupt/Event Selection

Any configurable line can be configured as a software interrupt/event line. To generate a software interrupt, the following steps must be implemented.

1. Configure the mask bit for interrupt/event lines
2. Clear the corresponding request bit in the software interrupt register.
3. Set the corresponding request bit in the software interrupt register.
4. Clear the corresponding request bit in the software interrupt register.

4 Software Configuration Process

This section takes I2CA FIFO interrupt as an example to describe the software configuration process of G32R5xx interrupt:

1. “Interrupt_initModule();” disable the global interrupts and disable all interrupts.
2. “Interrupt_initVectorTable();” initialize the NVIC vector table.
3. “Interrupt_setPriorityGroup(INTERRUPT_PRIGROUP_PREEMPT_7_6_SUB_5_0);” set the interrupt priority group.
4. “Interrupt_register(INT_I2CA_FIFO, &INT_I2CA_FIFO_IRQHandler);” according to the interrupt vector number, write the entry address of relevant interrupt service function to the RAM area vector table to complete the registration of the interrupt service function. The meanings of the two parameters here are as follows:
   a. INT_I2CA_FIFO: INT_I2CA_FIFO interrupt vector sequence number, namely its corresponding position in the interrupt vector table.
   b. INT_I2CA_FIFO_IRQHandler: The interrupt service function of the I2CA FIFO interrupt vector sequence number, namely writing the entry address of INT_I2CA_FIFO_IRQHandler function to the corresponding I2CA FIFO interrupt vector sequence number position in the RAM area vector table.
5. “Interrupt_setPriority(INT_I2CA_FIFO,1,0);” set the interrupt priority.
6. “Interrupt_enable(INT_I2CA_FIFO);” enable NVIC interrupt.

The example codes are as follows:

```
//
    // Initialize NVIC and clears NVIC registers. Disables CPU interrupts.
    //
Interrupt_initModule();
    //
    // Initialize the NVIC vector table with pointers to the shell
Interrupt
    // Service Routines (ISR).
    //
    Interrupt_initVectorTable();
    //
    // Interrupts that are used in this example are re-mapped to ISR
functions
    // found within this file.
    //
    Interrupt_register(INT_I2CA_FIFO, &INT_I2CA_FIFO_IRQHandler);
```

```
    //
    // Set the priority group to indicate the PREEMPT and SUB priority
bits.
    //
    Interrupt_setPriorityGroup(INTERRUPT_PRIGROUP_PREEMPT_7_6_SUB_5_0);
    //
    // Set the global and group priority to allow CPU interrupts
    // with higher priority
    //
    Interrupt_setPriority(INT_I2CA_FIFO,1,0);
    //
    // Enable interrupts required for this example
    //
Interrupt_enable(INT_I2CA_FIFO);
    //
    // Enable Global Interrupt (INTM) and realtime interrupt (DBGM)
    //
EINT;

ERTM;
```

5 Revision

Table 2 Document Revision History

Date	Version	Change History
January, 2025	1.0	New

Statement

This document is formulated and published by Geehy Semiconductor Co., Ltd. (hereinafter referred to as “Geehy”). The contents in this document are protected by laws and regulations of trademark, copyright and software copyright. Geehy reserves the right to make corrections and modifications to this document at any time. Read this document carefully before using Geehy products. Once you use the Geehy product, it means that you (hereinafter referred to as the “users”) have known and accepted all the contents of this document. Users shall use the Geehy product in accordance with relevant laws and regulations and the requirements of this document.

1. Ownership

This document can only be used in connection with the corresponding chip products or software products provided by Geehy. Without the prior permission of Geehy, no unit or individual may copy, transcribe, modify, edit or disseminate all or part of the contents of this document for any reason or in any form.

The “极海” or “Geehy” words or graphics with “®” or “™” in this document are trademarks of Geehy. Other product or service names displayed on Geehy products are the property of their respective owners.

2. No Intellectual Property License

Geehy owns all rights, ownership and intellectual property rights involved in this document.

Geehy shall not be deemed to grant the license or right of any intellectual property to users explicitly or implicitly due to the sale or distribution of Geehy products or this document.

If any third party’s products, services or intellectual property are involved in this document, it shall not be deemed that Geehy authorizes users to use the aforesaid third party’s products, services or intellectual property. Any information regarding the application of the product, Geehy hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party, unless otherwise agreed in sales order or sales contract.

3. Version Update

Users can obtain the latest document of the corresponding models when ordering Geehy products.

If the contents in this document are inconsistent with Geehy products, the agreement in the sales order or the sales contract shall prevail.

4. Information Reliability

The relevant data in this document are obtained from batch test by Geehy Laboratory or cooperative third-party testing organization. However, clerical errors in correction or errors caused by differences in testing environment may occur inevitably. Therefore, users should understand that Geehy does not bear any responsibility for such errors that may occur in this document. The relevant data in this document are only used to guide users as performance parameter reference and do not constitute Geehy's guarantee for any product performance.

Users shall select appropriate Geehy products according to their own needs, and effectively verify and test the applicability of Geehy products to confirm that Geehy products meet their own needs, corresponding standards, safety or other reliability requirements. If losses are caused to users due to user's failure to fully verify and test Geehy products, Geehy will not bear any responsibility.

5. Legality

USERS SHALL ABIDE BY ALL APPLICABLE LOCAL LAWS AND REGULATIONS WHEN USING THIS DOCUMENT AND THE MATCHING GEEHY PRODUCTS. USERS SHALL UNDERSTAND THAT THE PRODUCTS MAY BE RESTRICTED BY THE EXPORT, RE-EXPORT OR OTHER LAWS OF THE COUNTRIES OF THE PRODUCTS SUPPLIERS, GEEHY, GEEHY DISTRIBUTORS AND USERS. USERS (ON BEHALF OR ITSELF, SUBSIDIARIES AND AFFILIATED ENTERPRISES) SHALL AGREE AND PROMISE TO ABIDE BY ALL APPLICABLE LAWS AND REGULATIONS ON THE EXPORT AND RE-EXPORT OF GEEHY PRODUCTS AND/OR TECHNOLOGIES AND DIRECT PRODUCTS.

6. Disclaimer of Warranty

THIS DOCUMENT IS PROVIDED BY GEEHY "AS IS" AND THERE IS NO WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

GEEHY'S PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED FOR USE AS CRITICAL COMPONENTS IN MILITARY, LIFE-SUPPORT, POLLUTION CONTROL, OR HAZARDOUS SUBSTANCES MANAGEMENT SYSTEMS, NOR WHERE FAILURE COULD RESULT IN INJURY, DEATH, PROPERTY OR ENVIRONMENTAL DAMAGE.

IF THE PRODUCT IS NOT LABELED AS "AUTOMOTIVE GRADE," IT SHOULD NOT BE CONSIDERED SUITABLE FOR AUTOMOTIVE APPLICATIONS. GEEHY ASSUMES NO LIABILITY FOR THE USE BEYOND ITS SPECIFICATIONS OR GUIDELINES.

THE USER SHOULD ENSURE THAT THE APPLICATION OF THE PRODUCTS COMPLIES WITH ALL RELEVANT STANDARDS, INCLUDING BUT NOT LIMITED TO SAFETY, INFORMATION SECURITY, AND ENVIRONMENTAL REQUIREMENTS. THE USER ASSUMES FULL RESPONSIBILITY FOR THE SELECTION AND USE OF GEEHY PRODUCTS. GEEHY WILL BEAR NO RESPONSIBILITY FOR ANY DISPUTES ARISING FROM THE SUBSEQUENT DESIGN OR USE BY USERS.

7. Limitation of Liability

IN NO EVENT, UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL GEEHY OR ANY OTHER PARTY WHO PROVIDES THE DOCUMENT AND PRODUCTS "AS IS", BE LIABLE FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, DIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE DOCUMENT AND PRODUCTS (INCLUDING BUT NOT LIMITED TO LOSSES OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY USERS OR THIRD PARTIES). THIS COVERS POTENTIAL DAMAGES TO PERSONAL SAFETY, PROPERTY, OR THE ENVIRONMENT, FOR WHICH GEEHY WILL NOT BE RESPONSIBLE.

8. Scope of Application

The information in this document replaces the information provided in all previous versions of the document.