南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Embedded System and Microcomputer Principle

LAB8 Watchdog -- IWDG

2023 Fall
wangq9@mail.sustech.edu.cn

CONTENTS

01

IWDG Description

1. IWDG Description
-- What is watchdog

- A watchdog timer, usually simplified as watchdog, is an electronic timer that used to detect the device fault and reset the device to recover it when there is a fault.
- It is widely used in embedded systems.
- There is a down counter in a watchdog timer. The system will restart when the counter counts to a reload value.
- In normal case, a system should refresh the counter of watchdog periodically to maintain the system not to be restart.
- If the system has fault and cannot refresh the counter periodically, it will count to reload value and restart the system.

1. IWDG Description
-- Watchdog of STM32F103

- The STM32F10xxx have two embedded watchdogs peripherals which offer a combination of high safety level, timing accuracy and flexibility of use.
- Both watchdog peripherals (Independent and Window) serve to detect and resolve malfunctions due to software failure, and to trigger system reset or an interrupt (window watchdog only) when the counter reaches a given timeout value.

1. IWDG Description
-- Watchdog of STM32F103(continued)

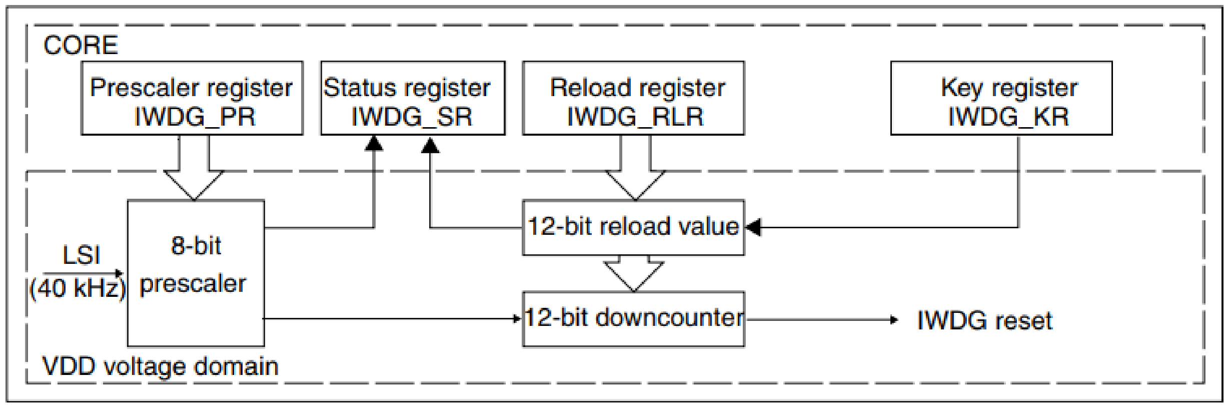
- The independent watchdog (**IWDG**) is clocked by a dedicated low-speed 40kHZ clock (**LSI**) and thus stays active even if the main clock fails.
- The IWDG is best suited to applications which require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints.
- The window watchdog (**WWDG**) clock is prescaled from the **APB1** clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.
- The WWDG is best suited to applications which require the watchdog to react within an accurate timing window.

1. IWDG Description
-- IWDG main features

- Independent watchdog (IWDG)
- Free-running downcounter
- clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Reset (if watchdog activated) when the downcounter value of 0x000 is reached

1. IWDG Description
-- IWDG functional description

- When the IWDG is started by writing the value 0xCCCC in the Key register(IWDG_KR), the counter starts counting down from the reset value of 0xFFF. When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).
- Whenever the key value 0xAAAA is written in the IWDG_KR register, the IWDG_RLR value is reloaded in the counter and the watchdog reset is prevented.

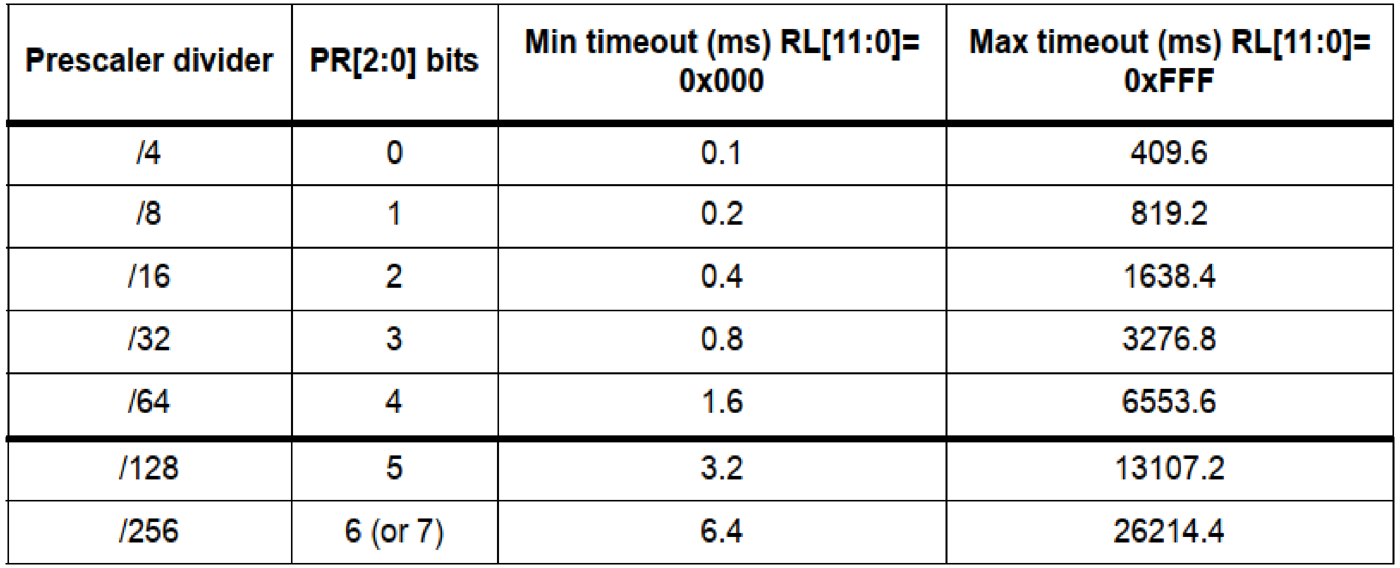


Independent watchdog block diagram

1. IWDG Description

-- IWDG timeout calculation

- T_{out} = presaler * reload / f_{LSI}
- f_{LSI} = 40kHz, presaler = 4 * $2^{PR[2:0]}$, reload = RLR
- Minimum timeout: one watchdog clock cycle
- Maximum timeout: (maximum value of IWDG_RLR register) * watchdog clock cycle



Prescaler divider	PR[2:0] bits	Min timeout (ms) RL[11:0]= 0x000	Max timeout (ms) RL[11:0]= 0xFFF
/4	0	0.1	409.6
/8	1	0.2	819.2
/16	2	0.4	1638.4
/32	3	0.8	3276.8
/64	4	1.6	6553.6
/128	5	3.2	13107.2
/256	6 (or 7)	6.4	26214.4

Min/max IWDG timeout period at 40kHz (LSI)

1. IWDG Description
-- IWDG configuration steps

- Cancel register write protection
- Set the prescaled coefficient of the independent watchdog to determine the clock
- Set the reload value to determine the overflow time
- Enable watchdog
- The application keeps refreshing (feeding the dog)

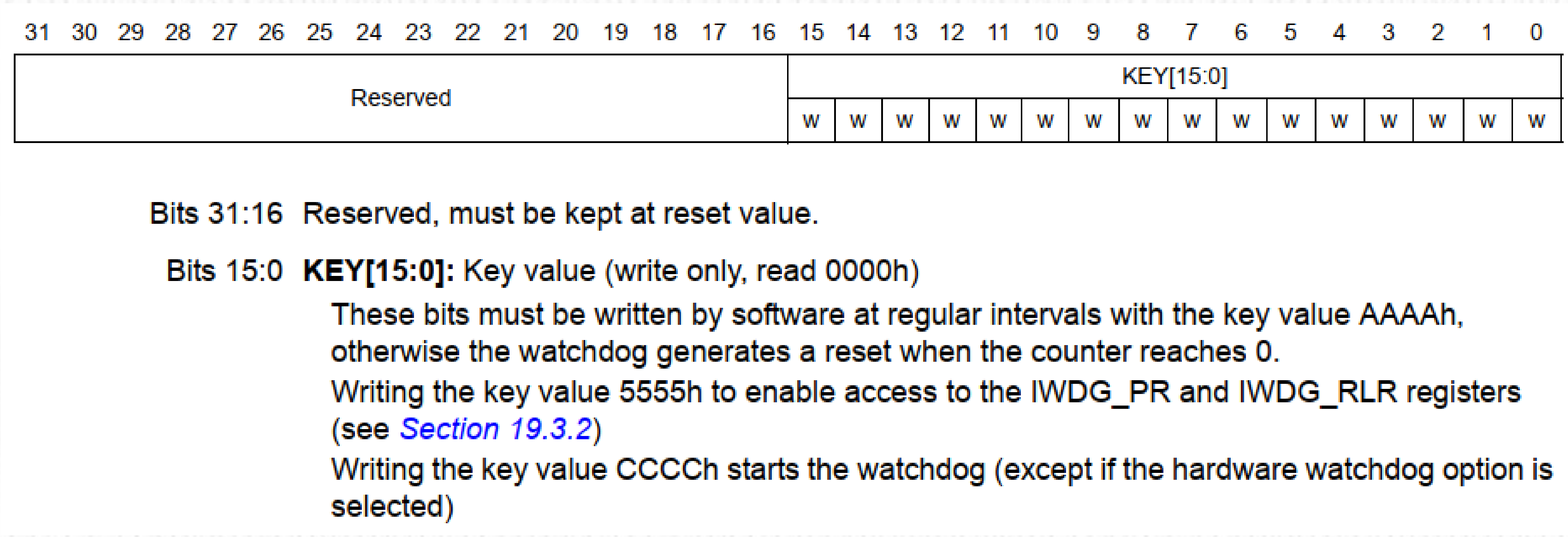
南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

02

IWDG Registers

2. IWDG Registers

- IWDG_KR: Key register



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																KEY[15:0]															
																w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **KEY[15:0]:** Key value (write only, read 0000h)

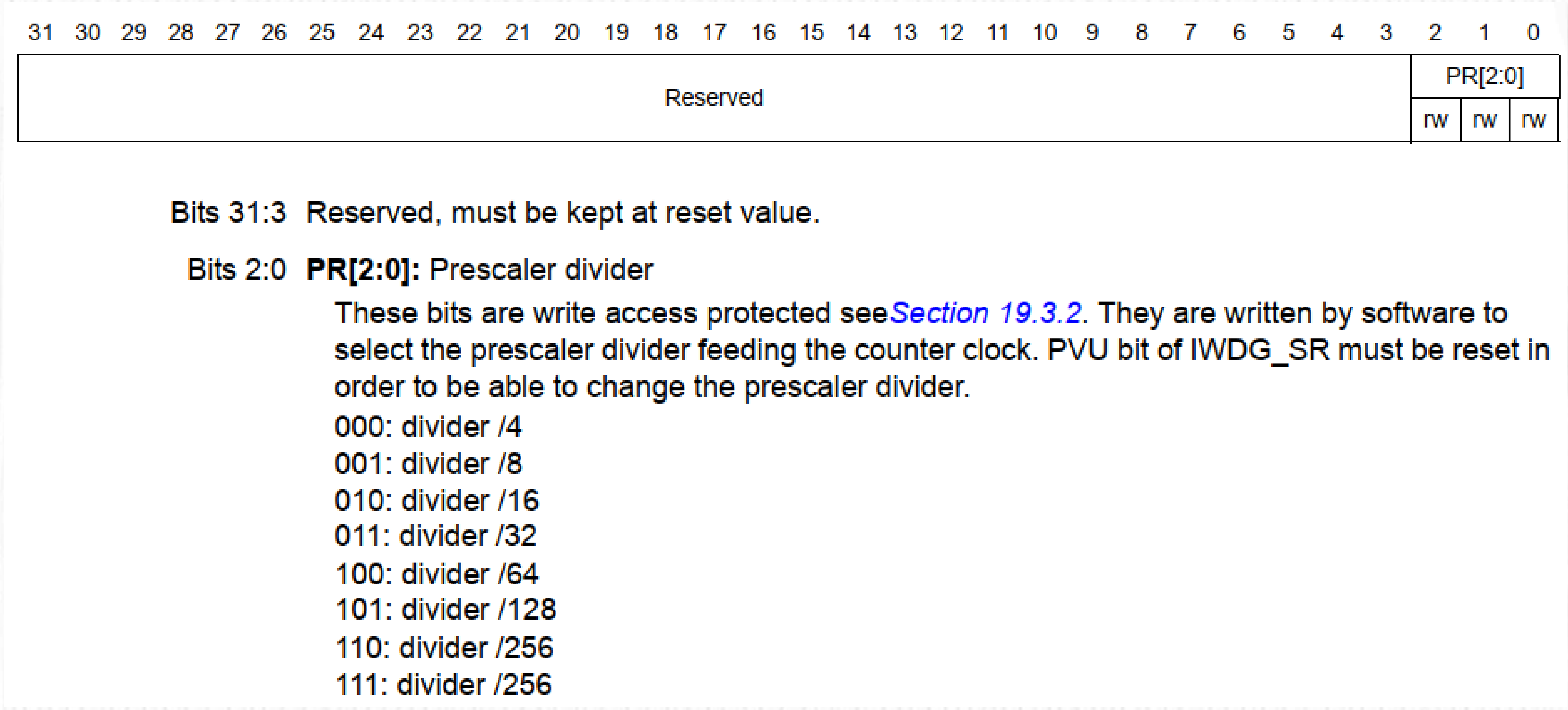
These bits must be written by software at regular intervals with the key value AAAAh, otherwise the watchdog generates a reset when the counter reaches 0.

Writing the key value 5555h to enable access to the IWDG_PR and IWDG_RLR registers (see *Section 19.3.2*)

Writing the key value CCCCh starts the watchdog (except if the hardware watchdog option is selected)

2. IWDG Registers

- IWDG_PR: Prescaler register



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													PR[2:0]		
																													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

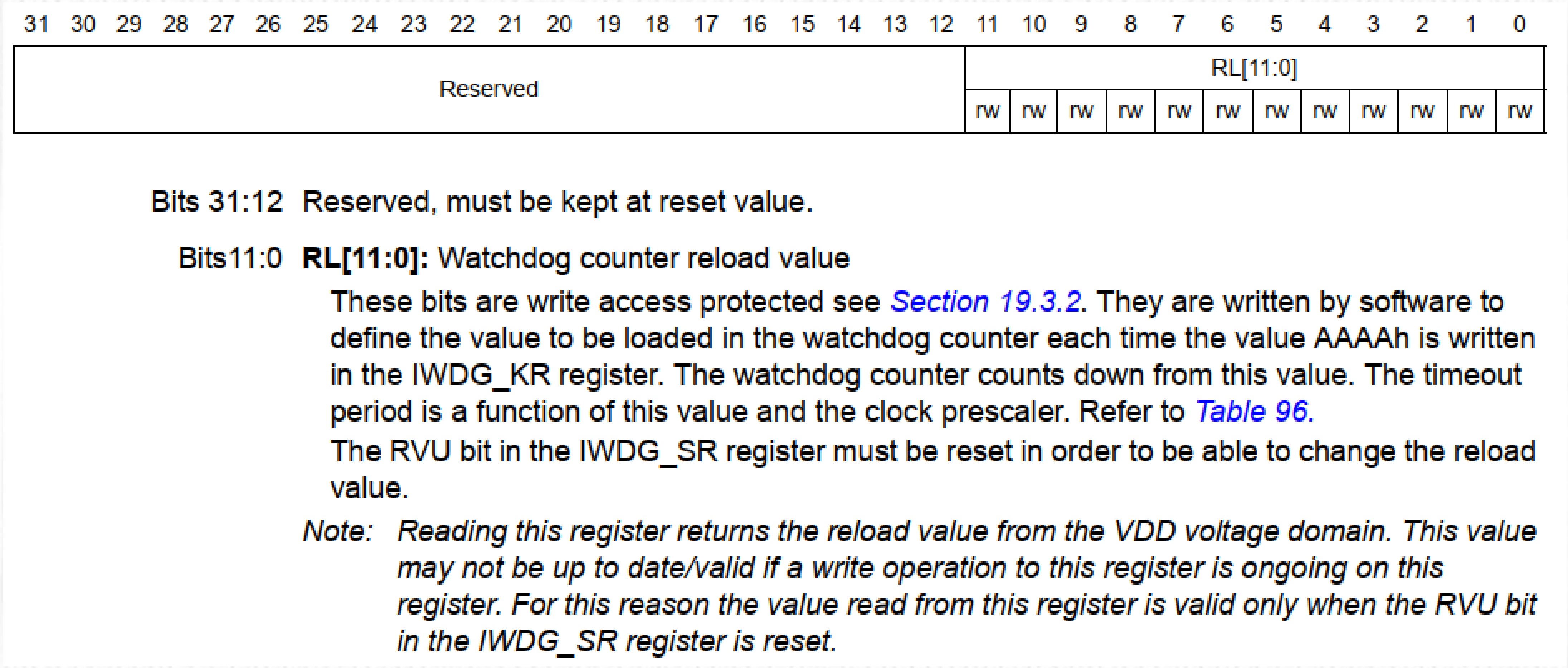
Bits 2:0 **PR[2:0]:** Prescaler divider

These bits are write access protected see*Section 19.3.2*. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of IWDG_SR must be reset in order to be able to change the prescaler divider.

000: divider /4
001: divider /8
010: divider /16
011: divider /32
100: divider /64
101: divider /128
110: divider /256
111: divider /256

2. IWDG Registers

- IWDG_RLR: Reload register



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				RL[11:0]											
																				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits11:0 **RL[11:0]:** Watchdog counter reload value

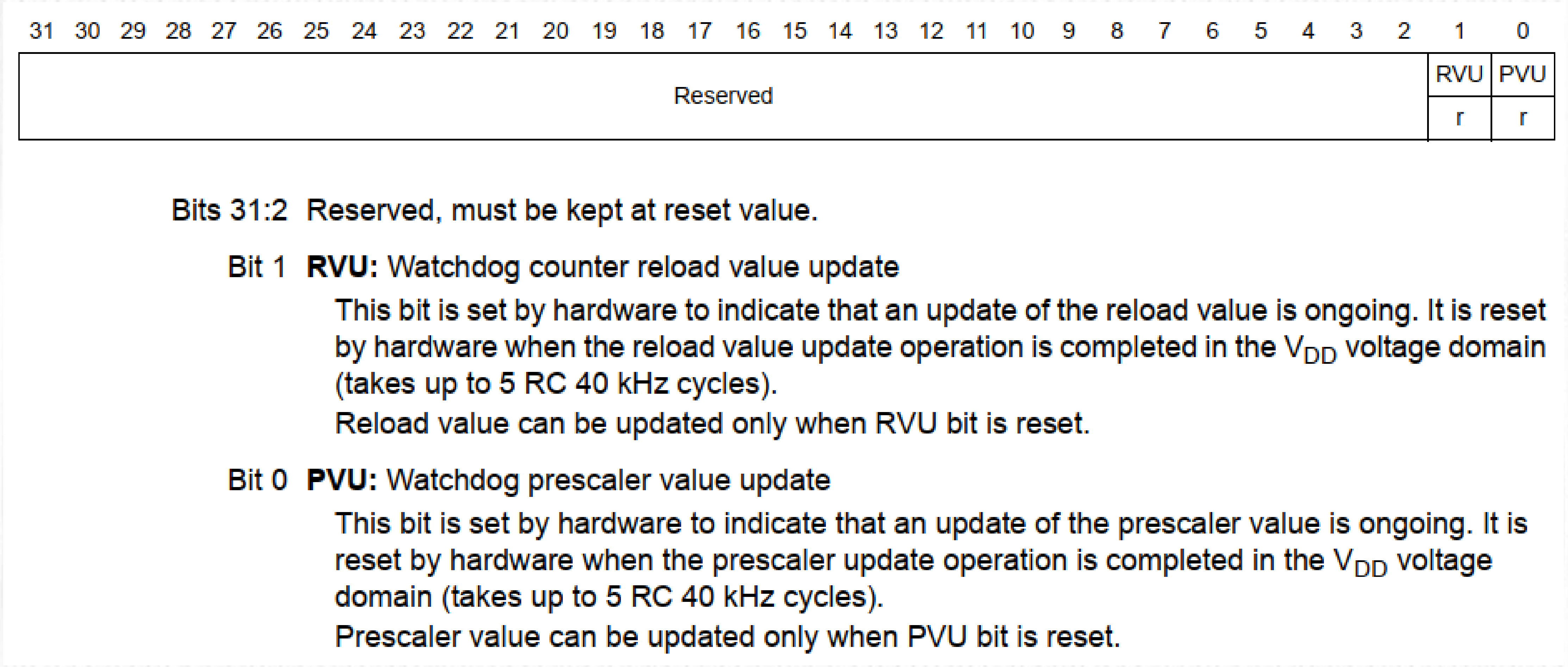
These bits are write access protected see *Section 19.3.2*. They are written by software to define the value to be loaded in the watchdog counter each time the value AAAAh is written in the IWDG_KR register. The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler. Refer to *Table 96.*

The RVU bit in the IWDG_SR register must be reset in order to be able to change the reload value.

Note: Reading this register returns the reload value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing on this register. For this reason the value read from this register is valid only when the RVU bit in the IWDG_SR register is reset.

2. IWDG Registers

- IWDG_SR: Status register



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														RVU	PVU
																														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **RVU:** Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to 5 RC 40 kHz cycles).

Reload value can be updated only when RVU bit is reset.

Bit 0 **PVU:** Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain (takes up to 5 RC 40 kHz cycles).

Prescaler value can be updated only when PVU bit is reset.

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

03

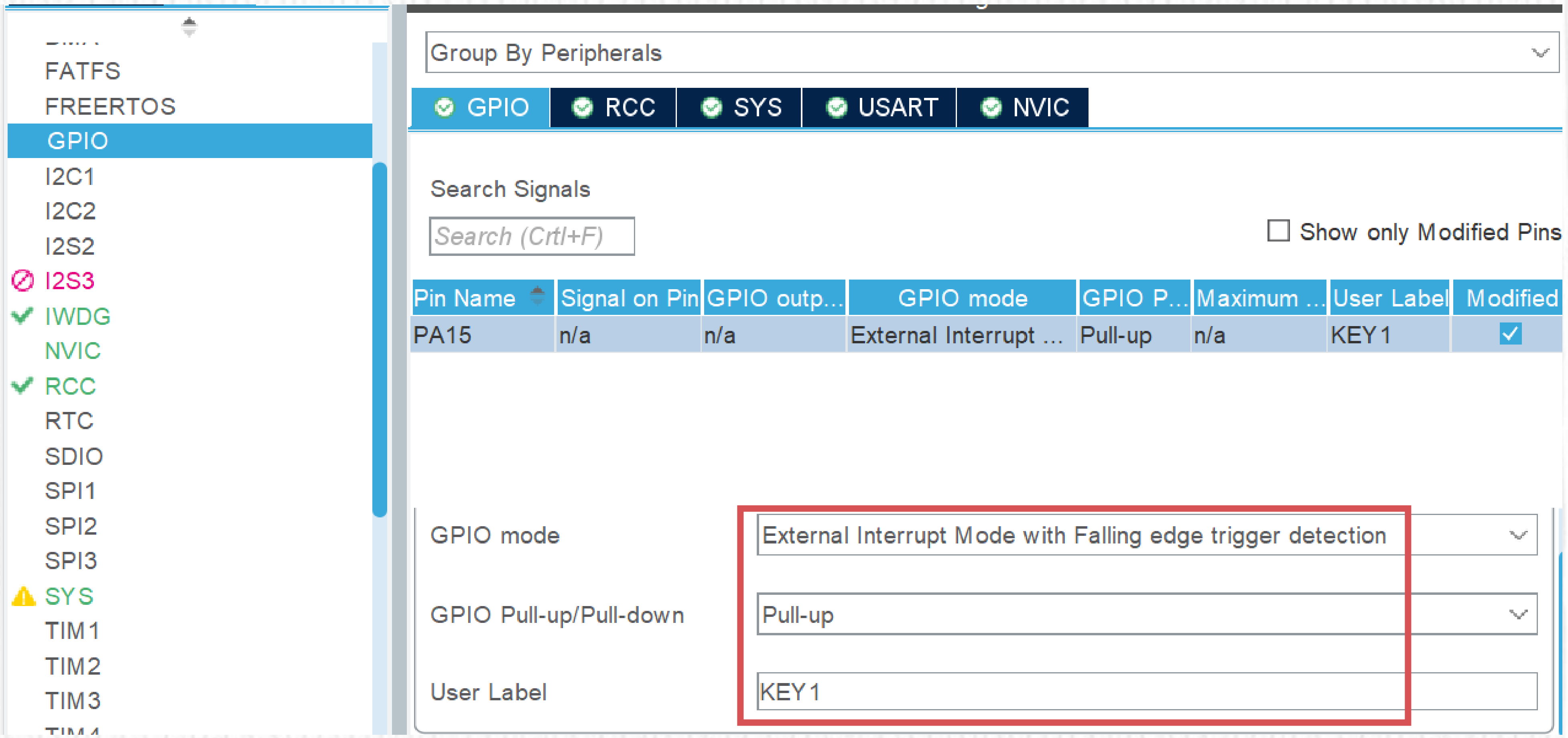
How to Program

3. How to Program

- Our Goal
  - Use KEY1 interrupt to refresh IWDG.
  - If the program is reset, the variable output restarts to count from 0.
  - If IWDG is refreshed before the downcounter has reached 0, the variable output counts increasingly by 1 each clock.

3. How to Program

- Configure GPIO
  - Set PA15(KEY1) as external interrupt source



FATFS
FREERTOS
GPIO
I2C1
I2C2
I2S2
I2S3
IWDG
NVIC
RCC
RTC
SDIO
SPI1
SPI2
SPI3
SYS
TIM1
TIM2
TIM3

Group By Peripherals

GPIO | RCC | SYS | USART | NVIC

Search Signals

Search (Crtl+F)

☐ Show only Modified Pins

Pin Name	Signal on Pin	GPIO outp...	GPIO mode	GPIO P...	Maximum ...	User Label	Modified
PA15	n/a	n/a	External Interrupt ...	Pull-up	n/a	KEY1	✓

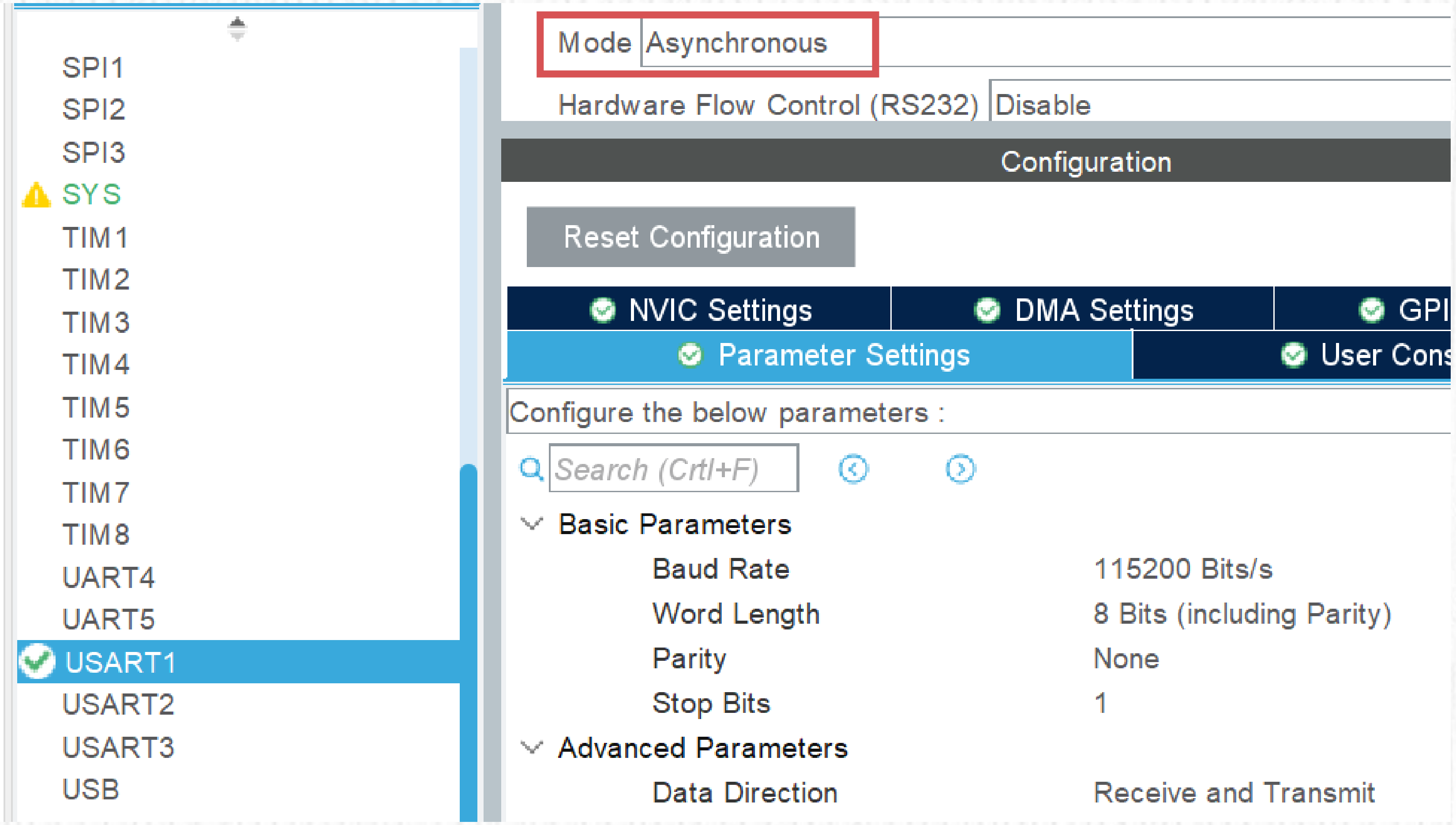
GPIO mode: External Interrupt Mode with Falling edge trigger detection

GPIO Pull-up/Pull-down: Pull-up

User Label: KEY1

3. How to Program

- Configure USART1
  - Set the USART1 as asynchronous mode



SPI1
SPI2
SPI3
SYS
TIM1
TIM2
TIM3
TIM4
TIM5
TIM6
TIM7
TIM8
UART4
UART5
USART1
USART2
USART3
USB

Mode	Asynchronous
Hardware Flow Control (RS232)	Disable

Configuration

Reset Configuration

NVIC Settings | DMA Settings | GPI
Parameter Settings | User Cons

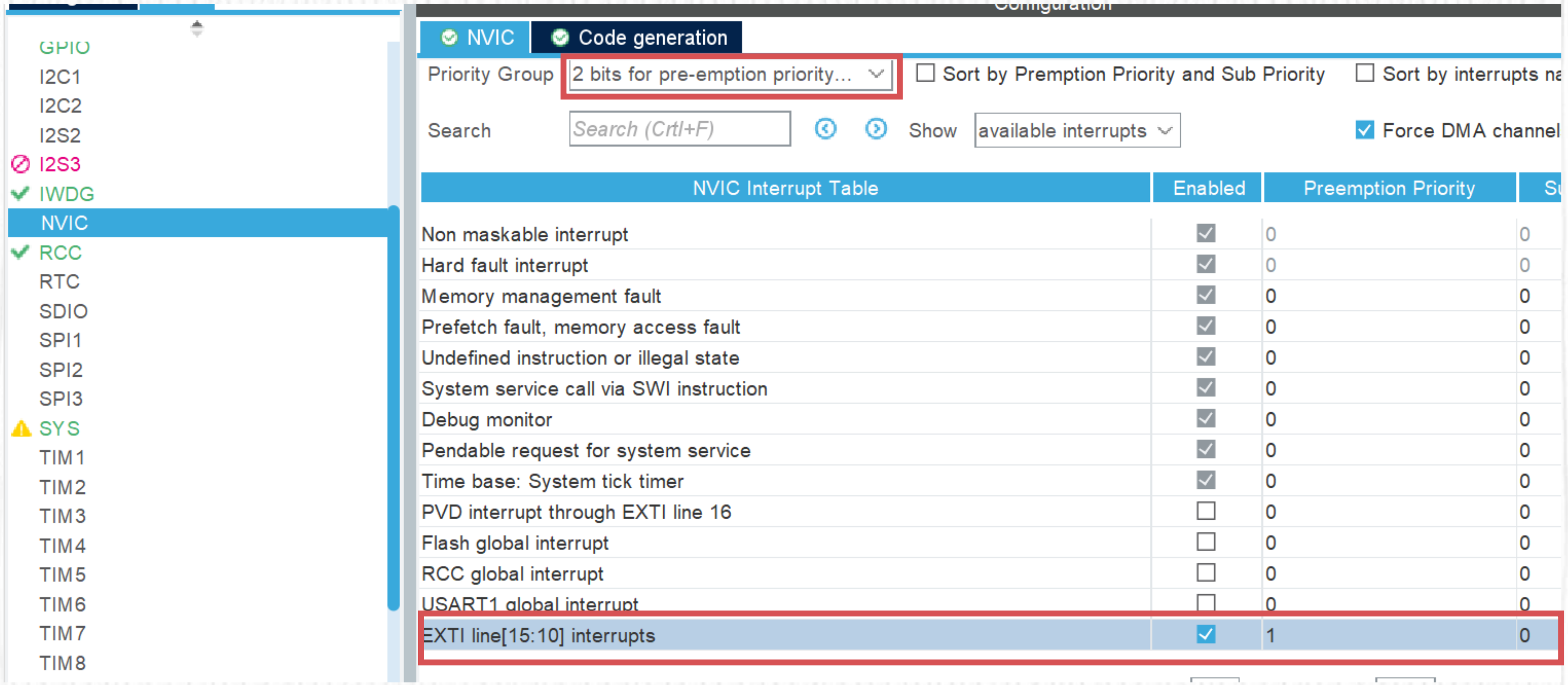
Configure the below parameters :

Search (Crtl+F)

Basic Parameters	
Baud Rate	115200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1
Advanced Parameters	
Data Direction	Receive and Transmit

3. How to Program

- Configure NVIC
  - Enable EXTI line[15:10] interrupt



GPIO
I2C1
I2C2
I2S2
I2S3
IWDG
NVIC
RCC
RTC
SDIO
SPI1
SPI2
SPI3
SYS
TIM1
TIM2
TIM3
TIM4
TIM5
TIM6
TIM7
TIM8

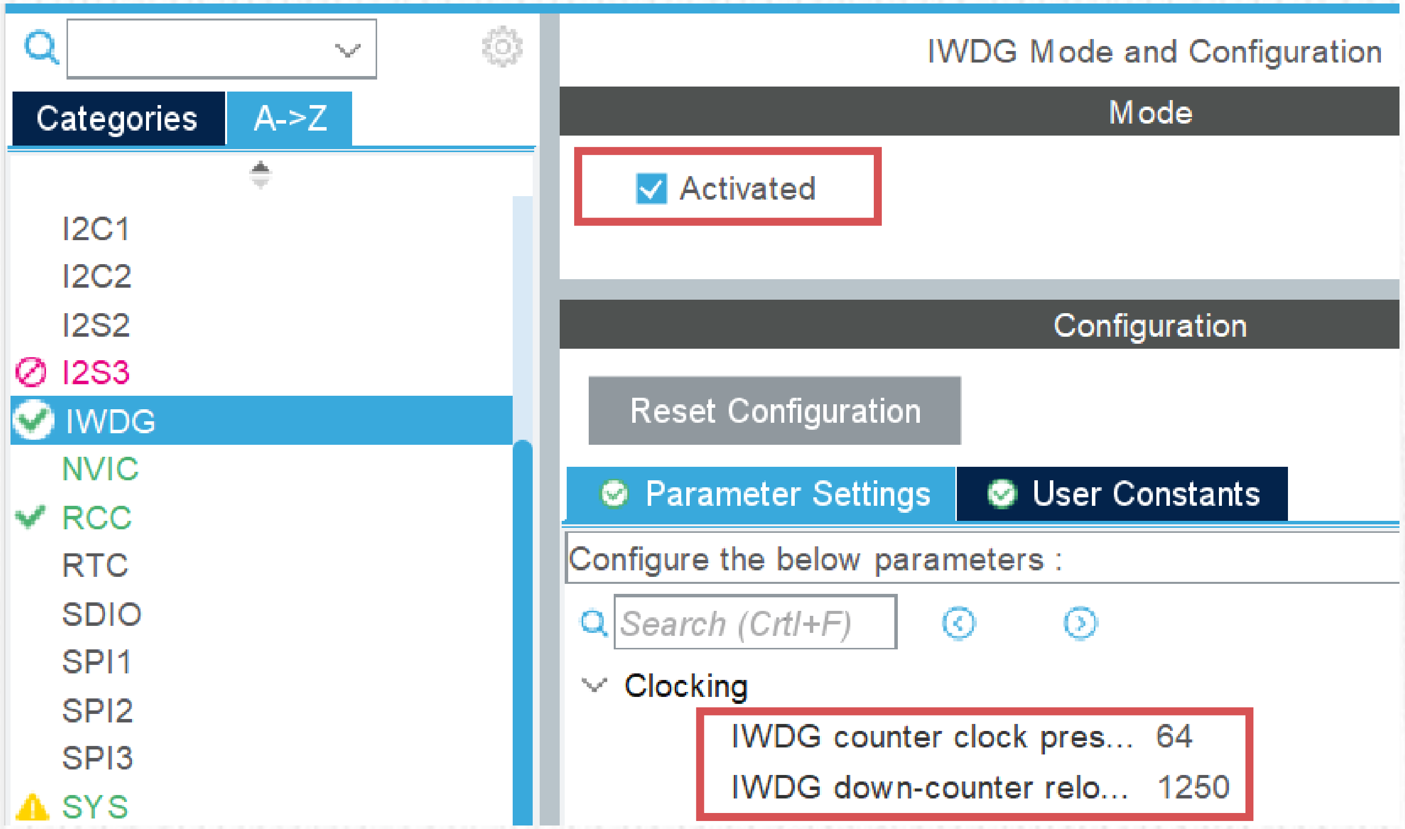
NVIC | Code generation

Priority Group: 2 bits for pre-emption priority...
☐ Sort by Premption Priority and Sub Priority
☐ Sort by interrupts na
Search: Search (Crtl+F)
Show: available interrupts
☑ Force DMA channel

NVIC Interrupt Table	Enabled	Preemption Priority	Su
Non maskable interrupt	✓	0	0
Hard fault interrupt	✓	0	0
Memory management fault	✓	0	0
Prefetch fault, memory access fault	✓	0	0
Undefined instruction or illegal state	✓	0	0
System service call via SWI instruction	✓	0	0
Debug monitor	✓	0	0
Pendable request for system service	✓	0	0
Time base: System tick timer	✓	0	0
PVD interrupt through EXTI line 16	☐	0	0
Flash global interrupt	☐	0	0
RCC global interrupt	☐	0	0
USART1 global interrupt	☐	0	0
EXTI line[15:10] interrupts	✓	1	0

3. How to Program

- Configure IWDG
  - Active the IWDG
  - Set the prescaler and reload value



Activated ✓

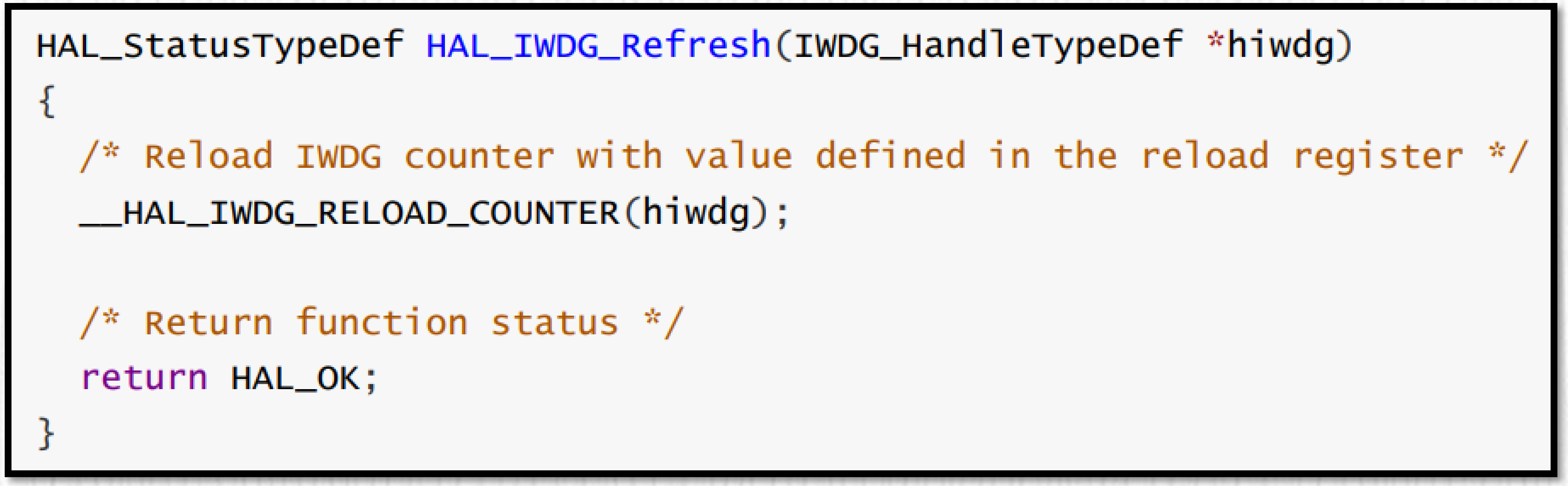
Configure the below parameters :

Clocking

IWDG counter clock pres...	64
IWDG down-counter relo...	1250

3. How to Program

- Some functions we used
  - Call HAL_IWDG_Refresh() function to refresh the IWDG (feed dog)



```
HAL_StatusTypeDef HAL_IWDG_Refresh(IWDG_HandleTypeDef *hiwdg)
{
  /* Reload IWDG counter with value defined in the reload register */
  __HAL_IWDG_RELOAD_COUNTER(hiwdg);

  /* Return function status */
  return HAL_OK;
}
```

3. How to Program

- Configure the external interrupt, and refresh the IWDG by pressing the KEY1 in **stm32f1xx_it.c**

```
extern IWDG_HandleTypeDef hiwdg;

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
	switch (GPIO_Pin) {
		case KEY1_Pin:
			if (HAL_GPIO_ReadPin(KEY1_GPIO_Port, KEY1_Pin) == GPIO_PIN_RESET){
				HAL_IWDG_Refresh(&hiwdg);
			}
			break;
		default:
			break;
	}
}
```

3. How to Program

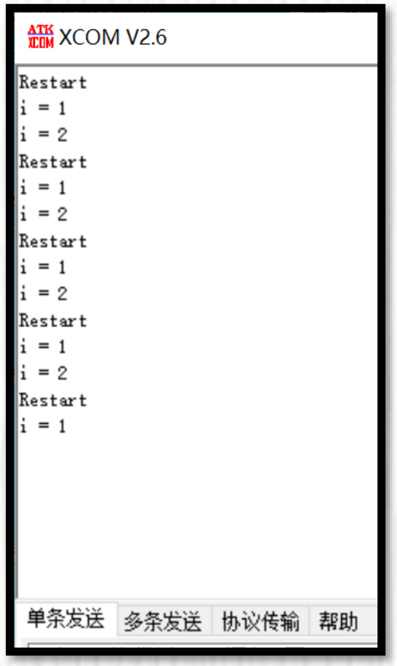
- Set a variable in **main.c** to show if the program is reset
- If the program is reset, the variable i will reset to 0.

```
int i = 0;
unsigned char msg[100];

HAL_UART_Transmit(&huart1, "Restart\r\n", 9, HAL_MAX_DELAY);
while (1)
{
		/* USER CODE END WHILE */
		/* USER CODE BEGIN 3 */
		i++;
		sprintf(msg, “i = %d\r\n", i);
		HAL_UART_Transmit(&huart1, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
		HAL_Delay(1000);
}
/* USER CODE END 3 */
```

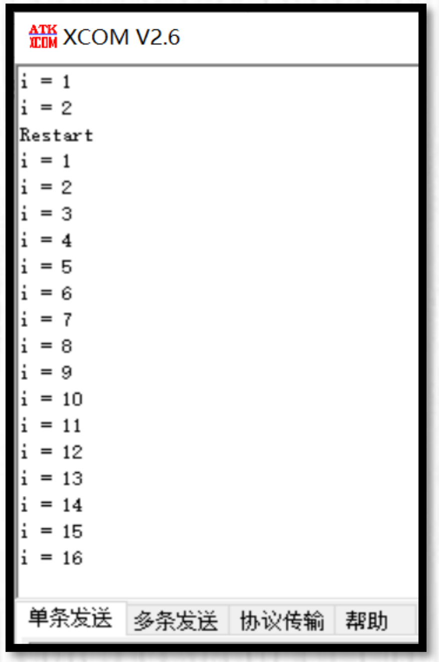
3. How to Program

- When the program is reset, the value of variable i comes back to 0, 1, 2 (shown as Fig.1).
- When press the KEY1 continuously, the value of variable i will increase (shown as Fig.2).



```
Restart
i = 1
i = 2
Restart
i = 1
i = 2
Restart
i = 1
i = 2
Restart
i = 1
i = 2
Restart
i = 1
```

Fig.1 The program resets periodically



```
i = 1
i = 2
Restart
i = 1
i = 2
i = 3
i = 4
i = 5
i = 6
i = 7
i = 8
i = 9
i = 10
i = 11
i = 12
i = 13
i = 14
i = 15
i = 16
```

Fig.2 The program runs continuously

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

04

Practice

4. Practice

- Run the IWDG demo on MiniSTM32 board.
- Tell what the timeout of IWDG is.
- Use the timer interrupt to refresh the IWDG in a suitable time interval instead of external interrupt of KEY1, and tell the parameters of IWDG and TIMER of your design.