Embedded System and Microcomputer Principle

LAB8 Watchdog -- WWDG

2022 Fall
wangq9@mail.sustech.edu.cn

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

CONTENTS

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

01

WWDG Description

1. WWDG Description
-- Watchdog of STM32F103

- STM32内置两个看门狗。
- **窗口看门狗（WWDG）**由从APB1时钟分频后得到时钟驱动。通过可配置的时间窗口来检测应用程序非正常的过迟或过早操作。WWDG适合那些要求看门狗在精确计时窗口起作用的程序。

1. WWDG Description
-- WWDG introduction

- 窗口看门狗通常被用来监测，由外部干扰或不可预见的逻辑条件造成的应用程序背离正常的运行序列而产生的软件故障。
- 除非递减计数器的值在T6位变成0前被刷新，看门狗电路在达到预置的时间周期时，会产生一个MCU复位。
- 在递减计数器达到窗口寄存器数值之前，如果7位的递减计数器数值(在控制寄存器中)被刷新，那么也将产生一个MCU复位。
- 递减计数器需要在一个有限的时间窗口中被刷新。

1. WWDG Description
-- WWDG main features

- Window watchdog (WWDG)
- 可编程的自由运行递减计数器
- 条件复位
  - 当递减计数器的值小于0x40，（若看门狗被启动）则产生复位。
  - 当递减计数器在窗口外被重新装载，（若看门狗被启动）则产生复位。
- 如果启动了看门狗并且允许中断，当递减计数器等于0x40时产生早期唤醒中断(EWI)，它可以被用于重装载计数器以避免WWDG复位。

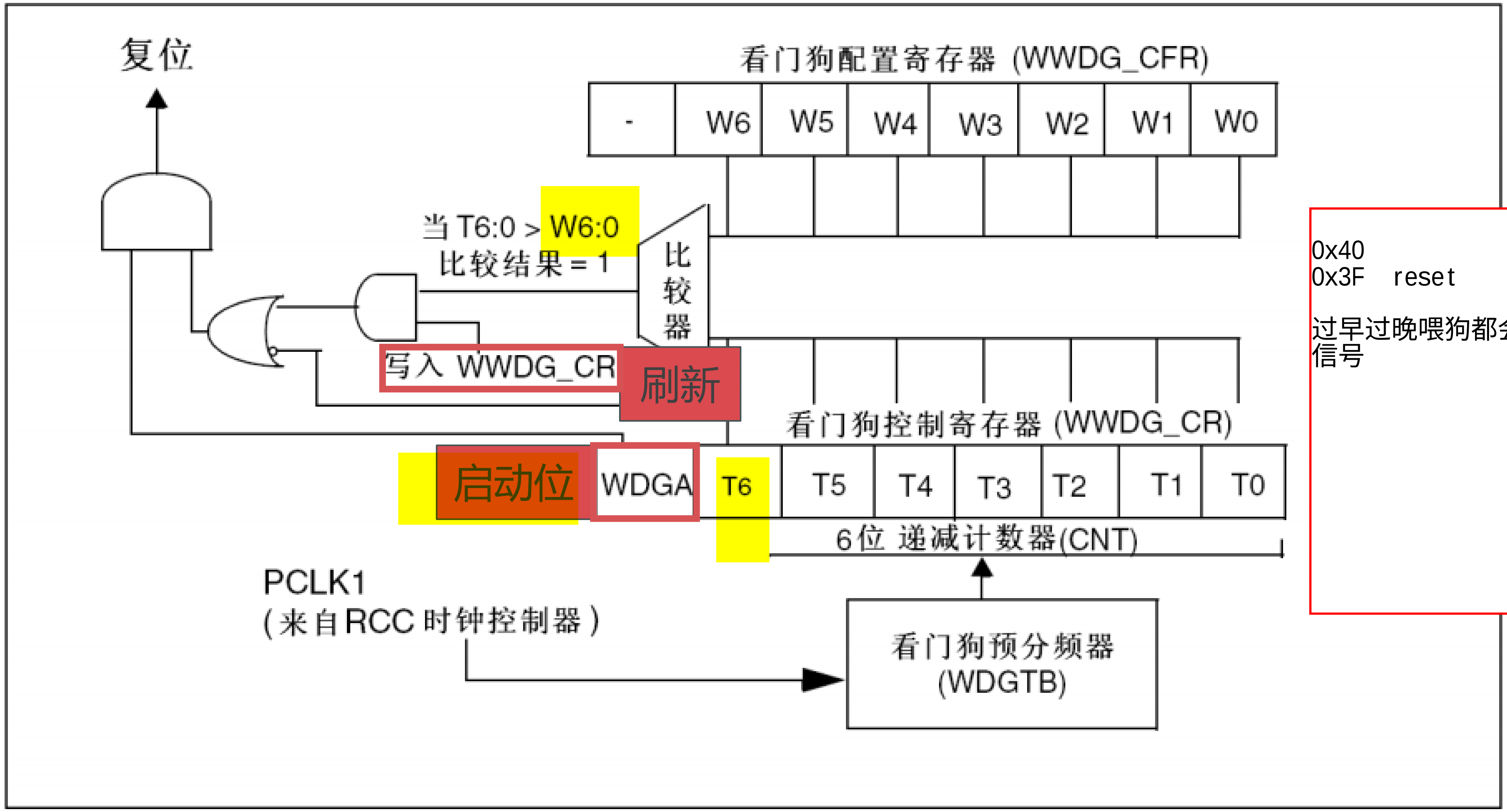
1. WWDG Description
-- WWDG functional description

- 看门狗被启动，递减计数器从0x40翻转到0x3F时，产生一个复位。
- 软件在计数器值大于窗口寄存器中的数值时重新装载计数器，将产生一个复位。
- 应用程序在正常运行过程中必须定期地写入WWDG_CR寄存器以防止MCU发生复位。
- 只有当计数器值小于窗口寄存器的值时，才能进行写操作。
- 储存在WWDG_CR寄存器中的数值必须在0xFF和0xC0之间：
  - 启动看门狗：在系统复位后，看门狗总是处于关闭状态，设置WWDG_CR寄存器的WDGA位能够开启看门狗，随后它不能再被关闭，除非发生复位。
  - 控制递减计数器

1. WWDG Description
-- WWDG functional description(continued)

- 控制递减计数器
  - 递减计数器处于自由运行状态，即使看门狗被禁止，递减计数器仍继续递减计数。
  - 当看门狗被启用时，T6位必须被设置，以防止立即产生一个复位。
  - T[5:0]位包含了看门狗产生复位之前的计时数目；复位前的延时时间在一个最小值和一个最大值之间变化，这是因为写入WWDG_CR寄存器时，预分频值是未知的。
  - WWDG_CFR中包含窗口的上限值：要避免产生复位，递减计数器必须在其值小于窗口寄存器的数值并且大于0x3F时被重新装载。
  - 另一个重装载计数器的方法是利用早期唤醒中断(EWI)。设置WWDG_CFR寄存器中的EWI位开启该中断。当递减计数器到达0x40时，则产生此中断，相应的中断服务程序(ISR)可以用来加载计数器以防止WWDG复位。在WWDG_SR寄存器中写’0’可以清除该中断。

1. WWDG Description
-- WWDG functional description(continued)



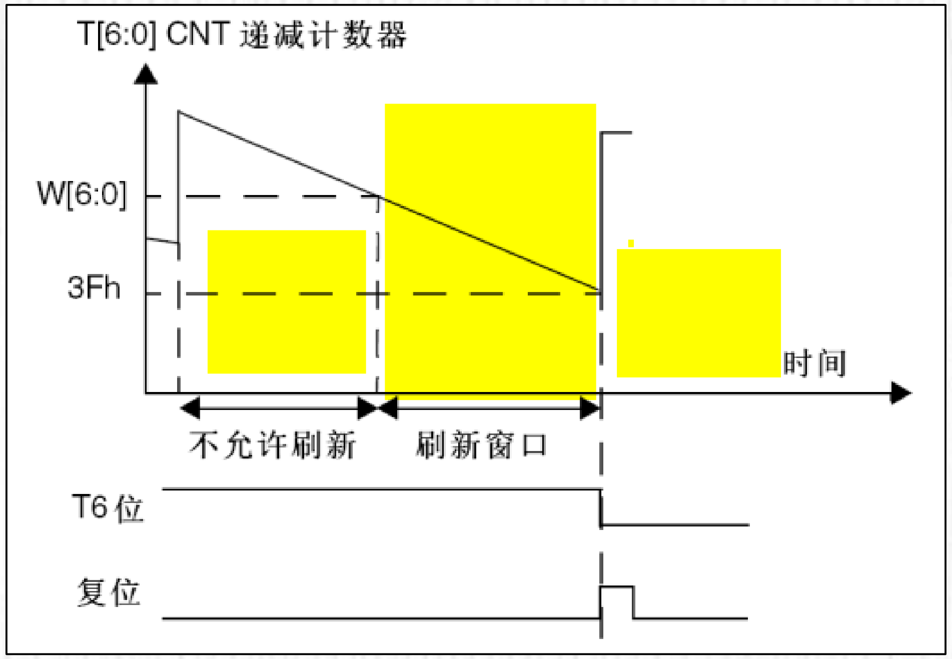
0x40
0x3F reset

过早过晚喂狗都会有复位信号

窗口看门狗框图

1. WWDG Description
-- WWDG timeout calculation

- $T_{WWDG} = T_{pclk1} * 4096 * 2^{WDGTB} * (T[5:0] + 1)$
  - T_{pclk1} : APB1时钟周期



窗口看门狗时序图

PCLK1 = 36MHz时的最小-最大超时值

Prescaler	WDGTB	Min timeout value	Max timeout value
1	0	113 µs	7.28 ms
2	1	227 µs	14.56 ms
4	2	455 µs	29.12 ms
8	3	910 µs	58.25 ms

1. WWDG Description
-- WWDG configuration steps

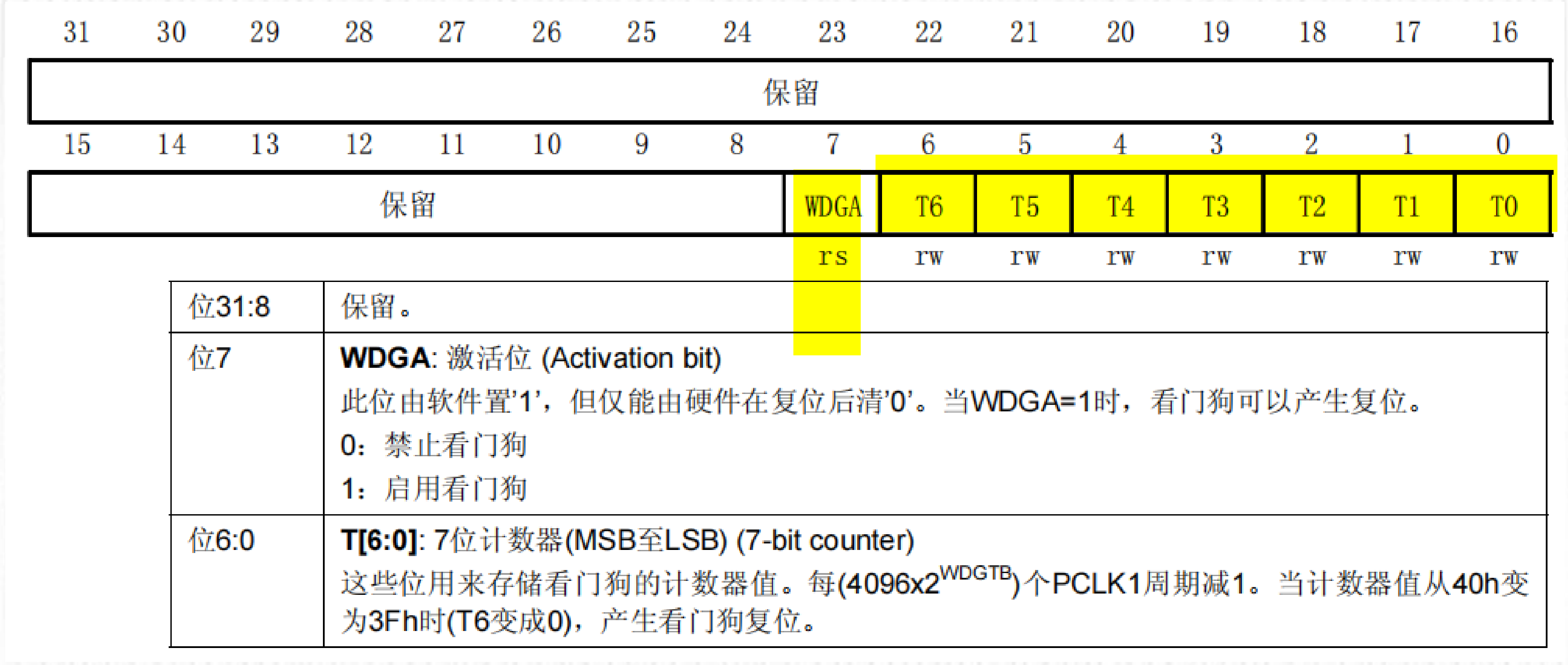
- 使能看门狗时钟
- 设置分频系数
- 设置上窗口值
- 开启提前唤醒中断并分组（可选）
- 使能看门狗
- 刷新看门狗T[6:0]的值（喂狗）
- 编写中断服务函数

02

WWDG Registers

2. WWDG Registers

- WWDG_CR: Control register
- 控制寄存器



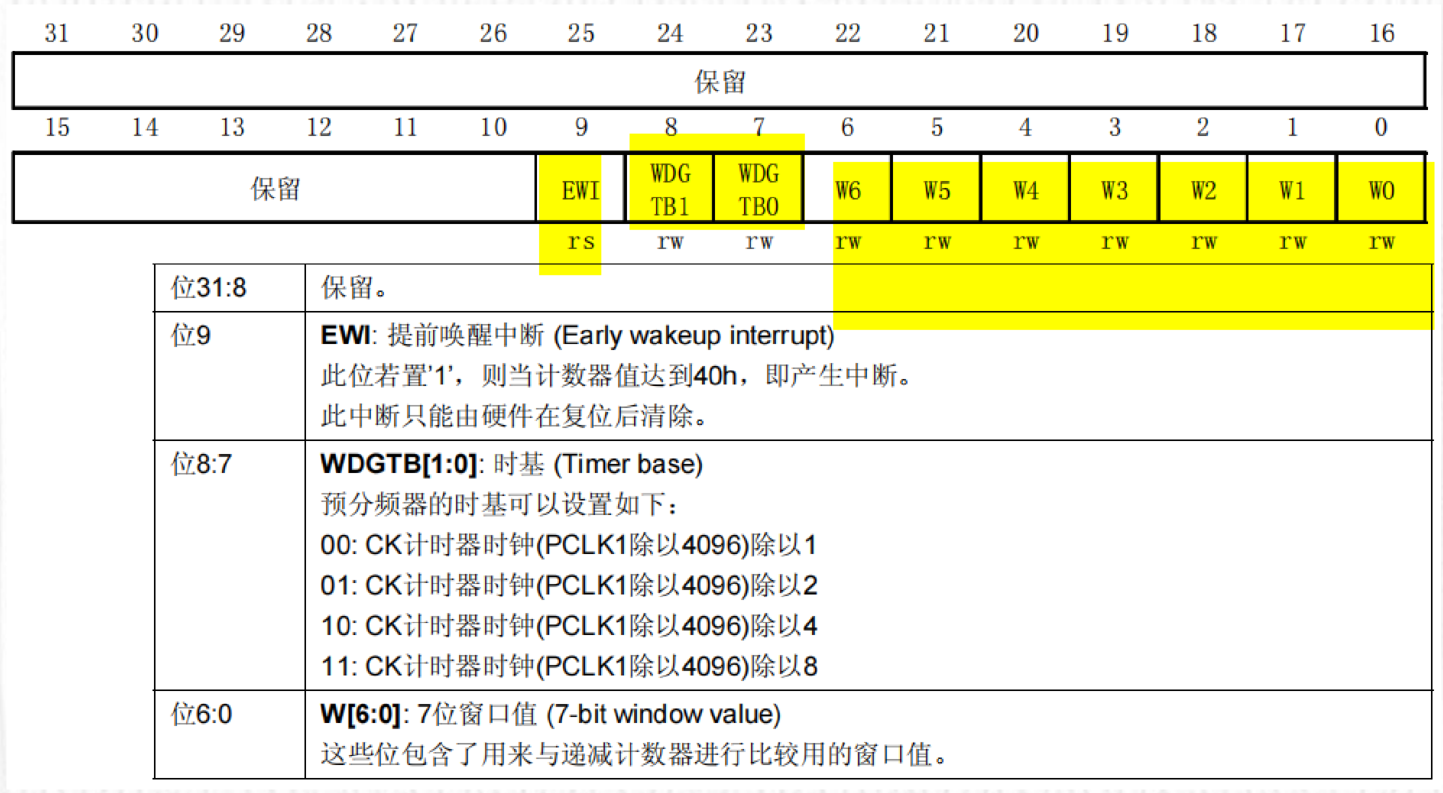
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								WDGA	T6	T5	T4	T3	T2	T1	T0
								rs	rw	rw	rw	rw	rw	rw	rw

位31:8	保留。
位7	**WDGA**: 激活位 (Activation bit) 此位由软件置'1'，但仅能由硬件在复位后清'0'。当WDGA=1时，看门狗可以产生复位。 0：禁止看门狗 1：启用看门狗
位6:0	**T[6:0]**: 7位计数器(MSB至LSB) (7-bit counter) 这些位用来存储看门狗的计数器值。每(4096x2^{WDGTB})个PCLK1周期减1。当计数器值从40h变为3Fh时(T6变成0)，产生看门狗复位。

2. WWDG Registers

- WWDG_CFR: Configuration register
- 配置寄存器



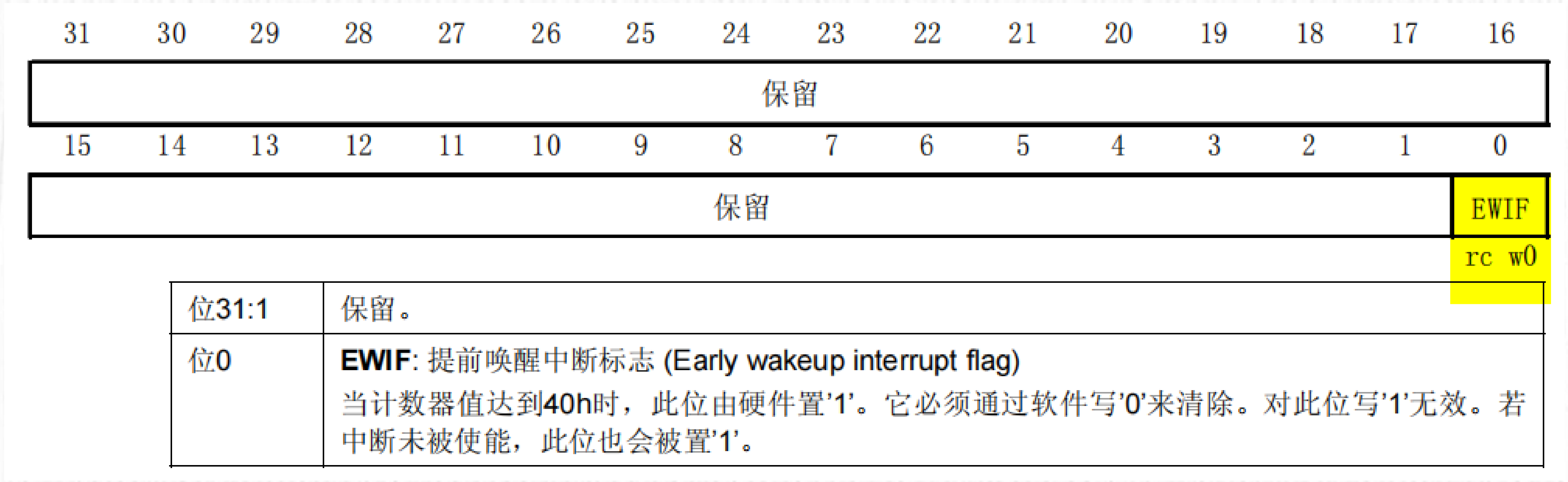
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						EWI	WDG TB1	WDG TB0	W6	W5	W4	W3	W2	W1	W0
						rs	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:8	保留。
位9	**EWI**: 提前唤醒中断 (Early wakeup interrupt) 此位若置'1'，则当计数器值达到40h，即产生中断。 此中断只能由硬件在复位后清除。
位8:7	**WDGTB[1:0]**: 时基 (Timer base) 预分频器的时基可以设置如下： 00: CK计时器时钟(PCLK1除以4096)除以1 01: CK计时器时钟(PCLK1除以4096)除以2 10: CK计时器时钟(PCLK1除以4096)除以4 11: CK计时器时钟(PCLK1除以4096)除以8
位6:0	**W[6:0]**: 7位窗口值 (7-bit window value) 这些位包含了用来与递减计数器进行比较用的窗口值。

2. WWDG Registers

- WWDG_SR: Status register
- 状态寄存器



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留															EWIF
															rc w0

位31:1	保留。
位0	**EWIF**: 提前唤醒中断标志 (Early wakeup interrupt flag) 当计数器值达到40h时，此位由硬件置’1’。它必须通过软件写’0’来清除。对此位写’1’无效。若中断未被使能，此位也会被置’1’。

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

03

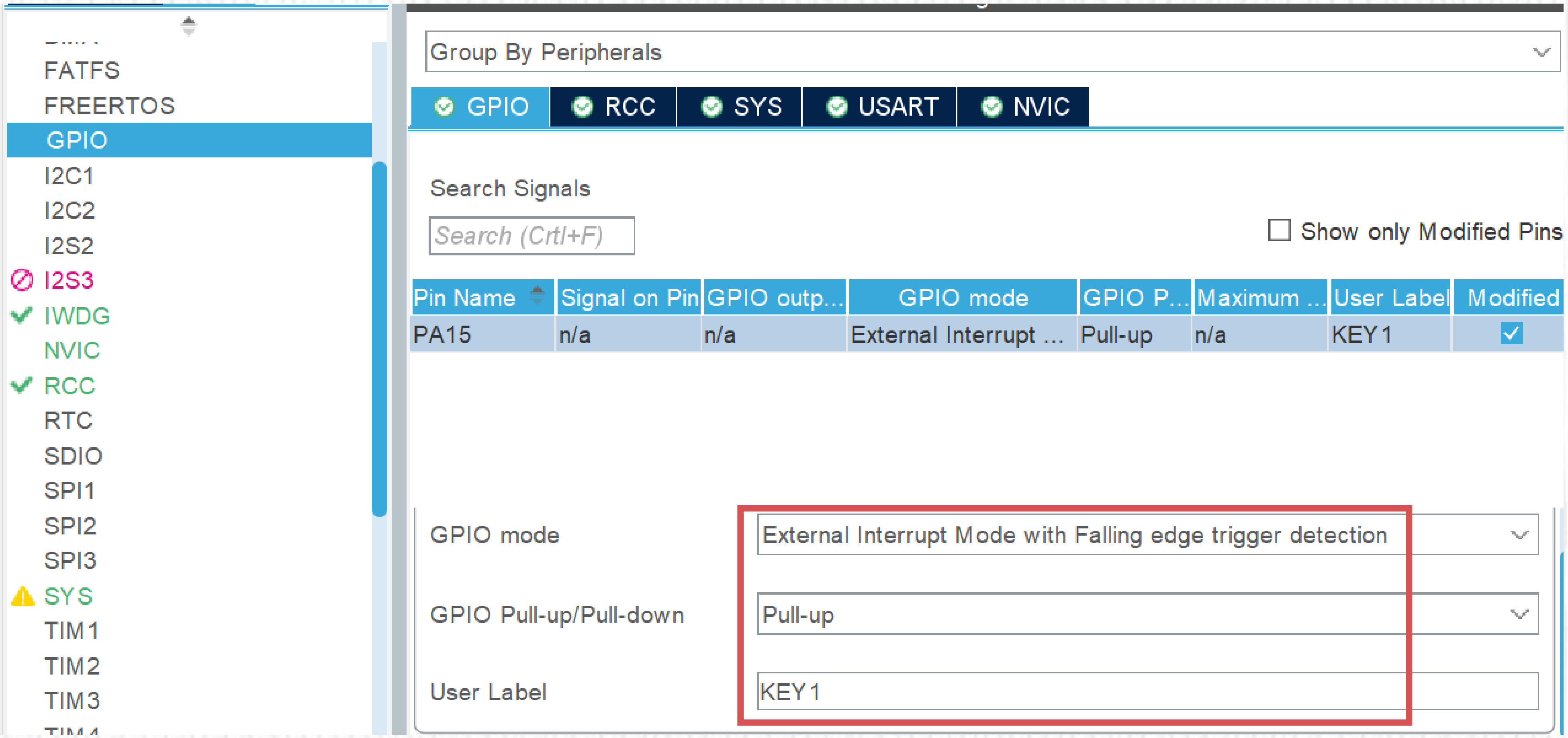
How to Program

3. How to Program

- Our Goal
  - Use EWI to refresh WWDG.
  - Use KEY1 interrupt to refresh WWDG.
  - If the program is reset, the variable output restarts to count from 0.
  - If WWDG is refreshed before the downcounter has reached the window register value, the variable output restarts to count from 0.
  - If WWDG is refreshed when the downcounter is between the window register value and 0x3F, the variable output counts increasingly by 1 each clock.

3. How to Program

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

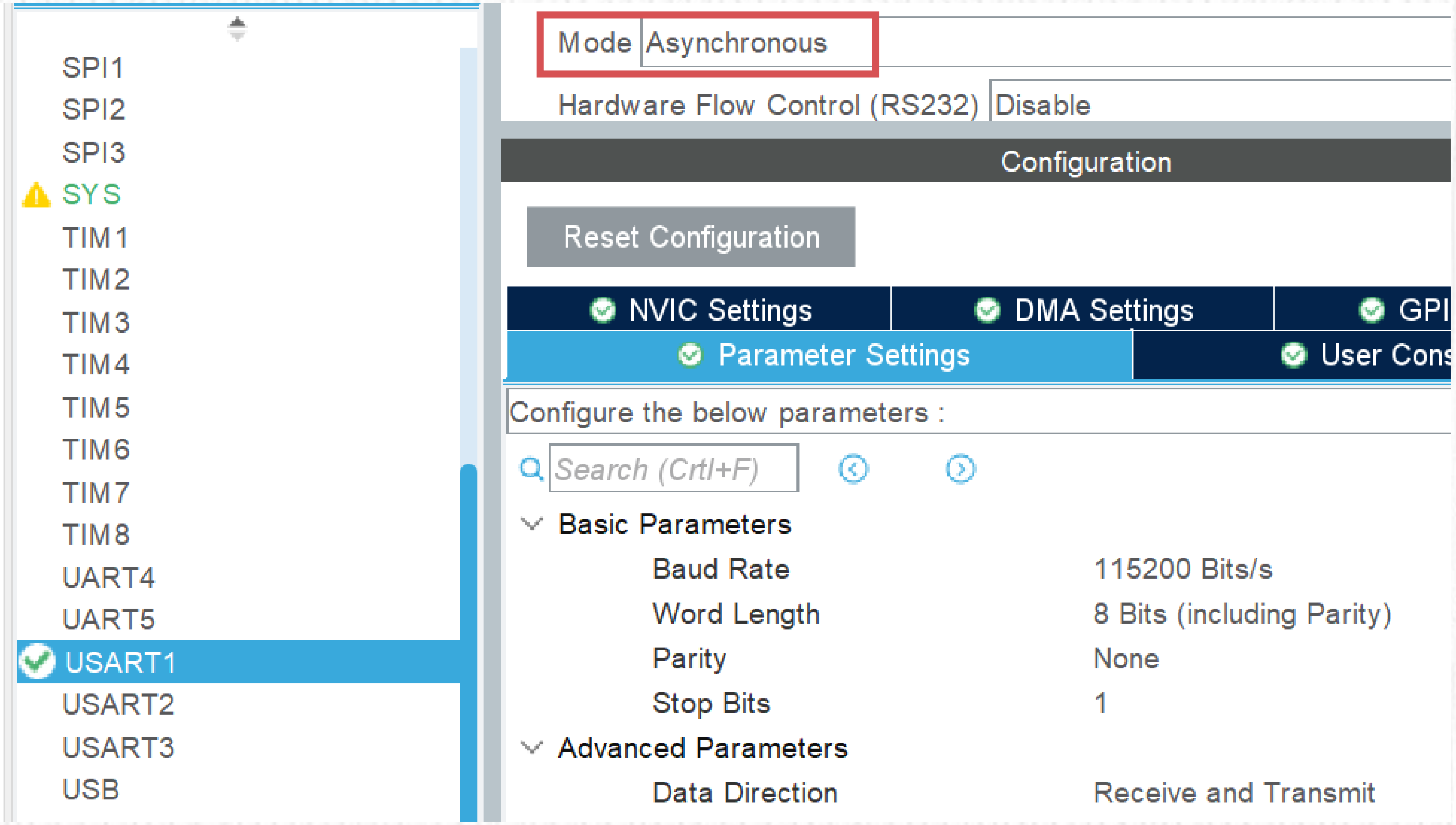


Pin Name	Signal on Pin	GPIO outp...	GPIO mode	GPIO P...	Maximum ...	User Label	Modified
PA15	n/a	n/a	External Interrupt ...	Pull-up	n/a	KEY1	✓

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



Mode Asynchronous

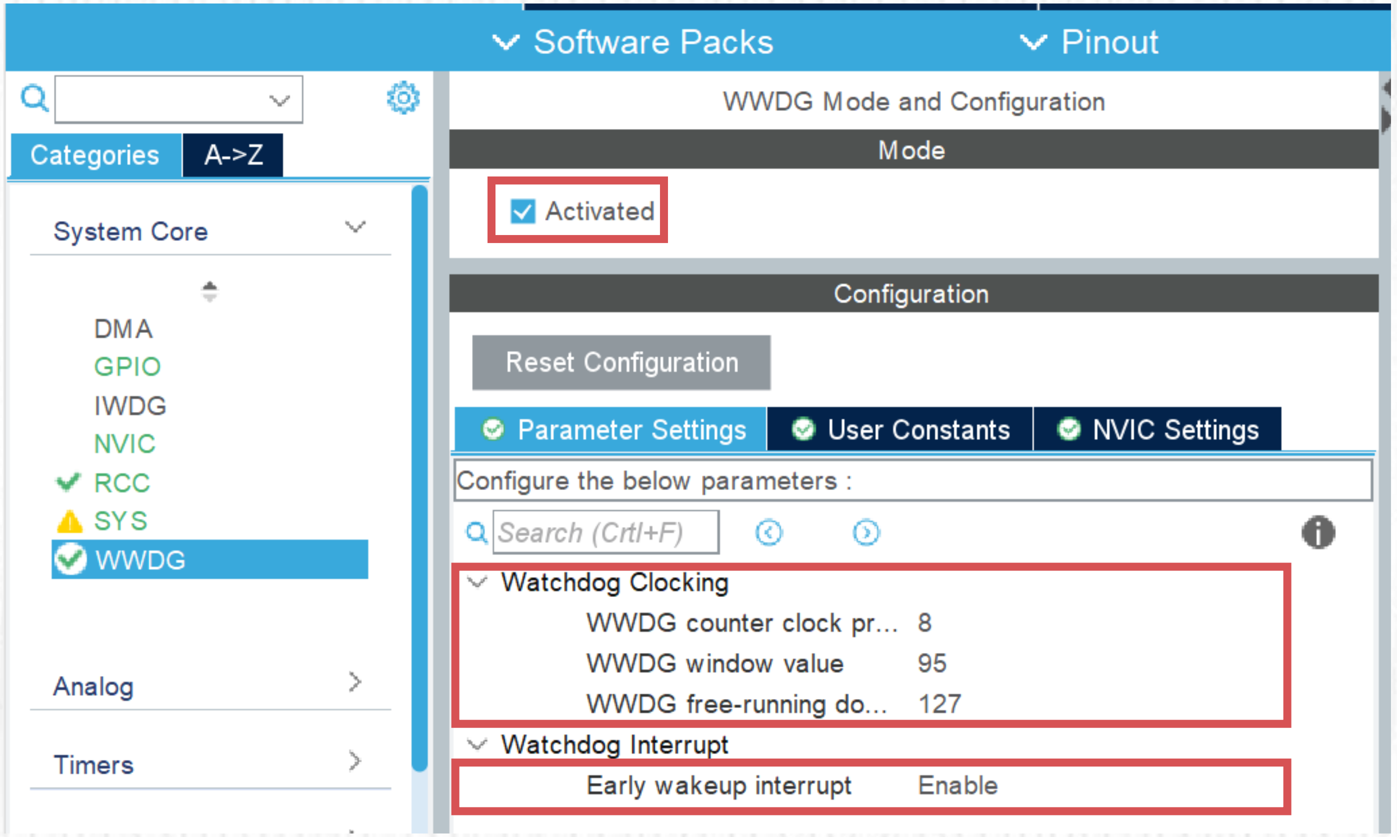
Hardware Flow Control (RS232) Disable

Configure the below parameters :

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 WWDG
  - Active the WWDG
  - Set the WWDG parameters, enable EWI



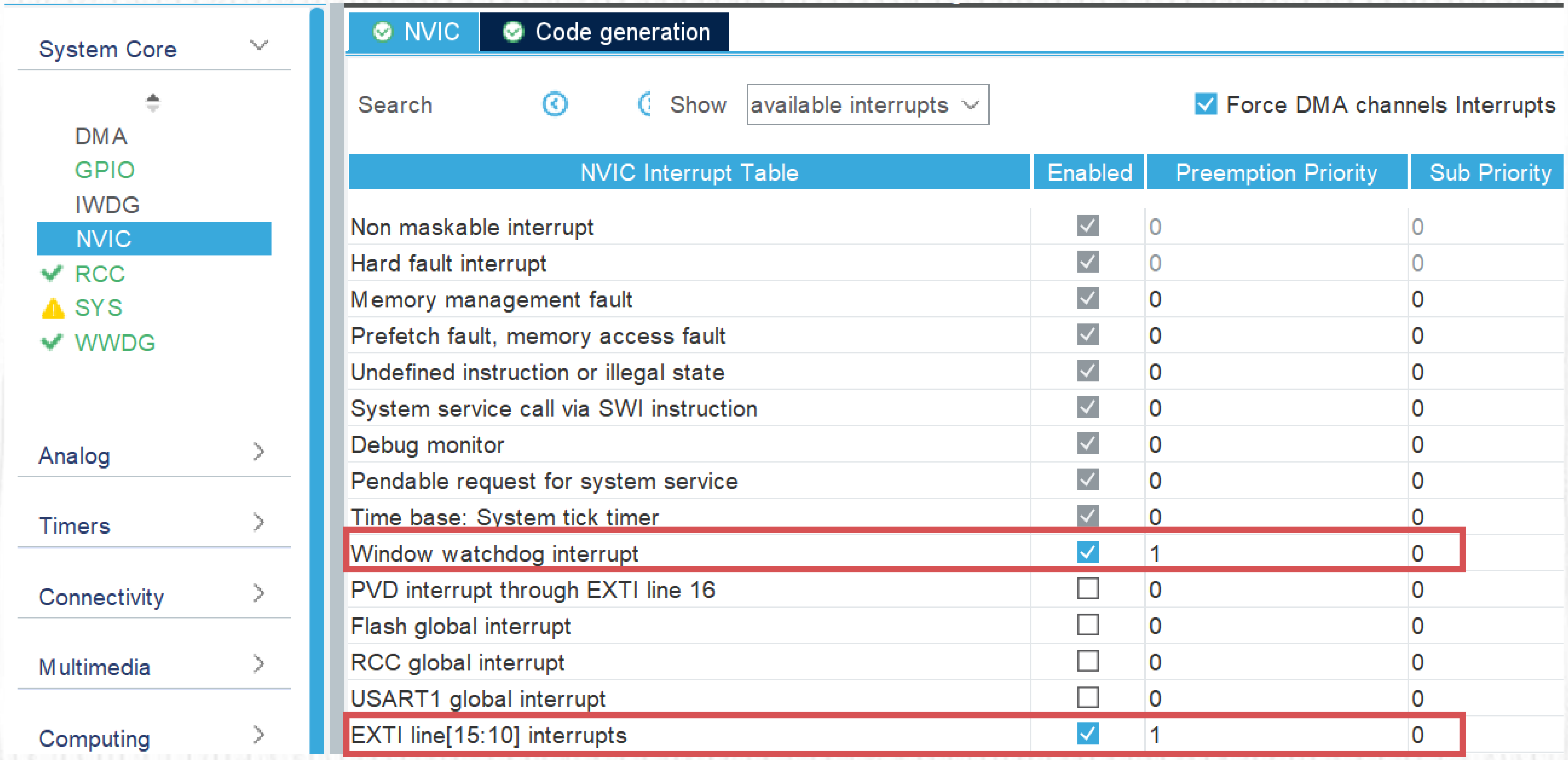
☑ Activated

Configure the below parameters :

Watchdog Clocking	
WWDG counter clock pr...	8
WWDG window value	95
WWDG free-running do...	127
Watchdog Interrupt	
Early wakeup interrupt	Enable

3. How to Program

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

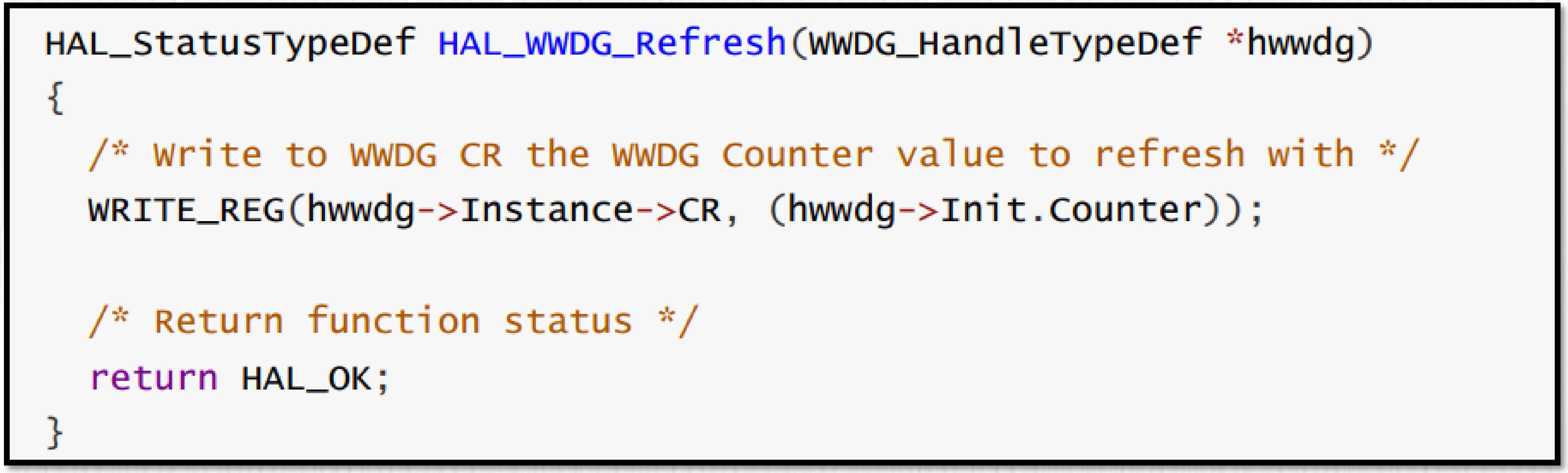


☑ Force DMA channels Interrupts

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
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
Window watchdog interrupt	✓	1	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

- Some functions we used
  - Call HAL_WWDG_Refresh() function to refresh the WWDG (feed dog)



```
HAL_StatusTypeDef HAL_WWDG_Refresh(WWDG_HandleTypeDef *hwwdg)
{
  /* Write to WWDG CR the WWDG Counter value to refresh with */
  WRITE_REG(hwwdg->Instance->CR, (hwwdg->Init.Counter));

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

3. How to Program

- When the counter is 0x40, the EWI will be triggered, it will call HAL_WWDG_EarlyWakeupCallback(), which is a weak function, we should re-implement it in **main.c or stm32f1xx_it.c**
- In this demo, we refresh the WWDG in EWI callback function, but in real case, we should record some critical information, for when EWI is triggered, the system has software fault already.
- Let the interrupt finish in 1 cycle, because the interrupt is triggered while the MCU is reset when the counter is 0x3F.

```
void HAL_WWDG_EarlyWakeupCallback(WWDG_HandleTypeDef* hwwdg)
{
		HAL_WWDG_Refresh(hwwdg);
}
```

3. How to Program

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

```
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_WWDG_Refresh(&hwwdg);
						}
						break;
				default:
						break;
		}
}
```

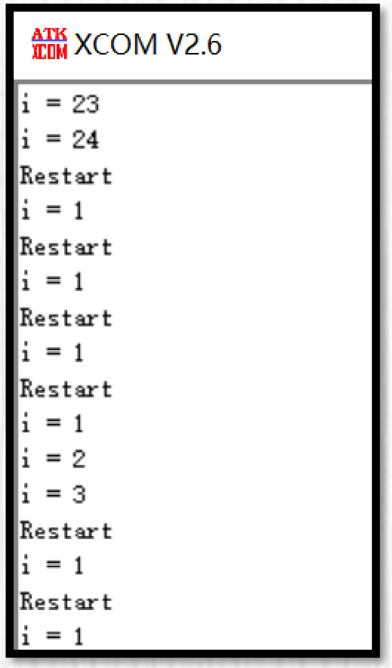
3. How to Program

- 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 */
```

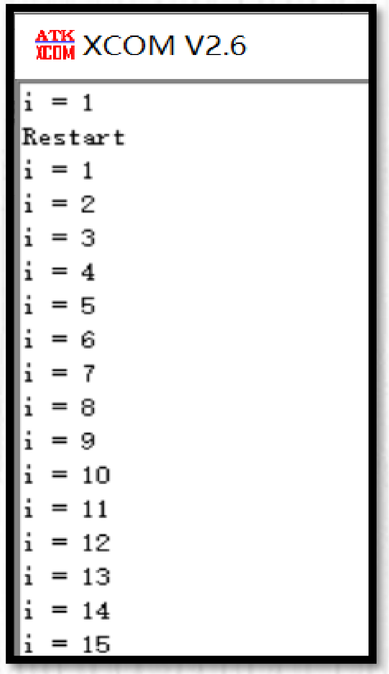
3. How to Program

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


XCOM V2.6

```
i = 23
i = 24
Restart
i = 1
Restart
i = 1
Restart
i = 1
Restart
i = 1
i = 2
i = 3
Restart
i = 1
Restart
i = 1
```

Fig.1 The program resets periodically


XCOM V2.6

```
i = 1
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
```

Fig.2 The program runs continuously

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

04

Practice

4. Practice

- Run the WWDG demo on MiniSTM32 board.
- Tell the time range in the demo to refresh WWDG without resetting it.
- Use the timer interrupt to refresh the WWDG in a suitable time interval instead of the pressing of KEY1, and tell the parameters of WWDG and TIMER of your design.