Embedded System and Microcomputer Principle

LAB9 Watchdog -- WWDG

2023 Fall

wangq9@mail.sustech.edu.cn

CONTENTS

01

WWDG Description

1. WWDG Description
-- Window Watchdog of STM32F103

- The STM32F10xxx have two embedded watchdogs peripherals which offer a combination of high safety level, timing accuracy and flexibility of use.
- 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. WWDG Description

-- WWDG introduction

- The window watchdog is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence.
- The watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the downcounter before the T6 bit becomes cleared.
- An MCU reset is also generated if the 7-bit downcounter value (in the control register) is refreshed before the downcounter has reached the window register value. This implies that the counter must be refreshed in a limited window.

1. WWDG Description

-- WWDG main features

- Window watchdog (WWDG)
- Programmable free-running downcounter
- Conditional reset
  - – Reset (if watchdog activated) when the downcounter value becomes less than 0x40
  - – Reset (if watchdog activated) if the downcounter is reloaded outside the window
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the downcounter is equal to 0x40.

1. WWDG Description

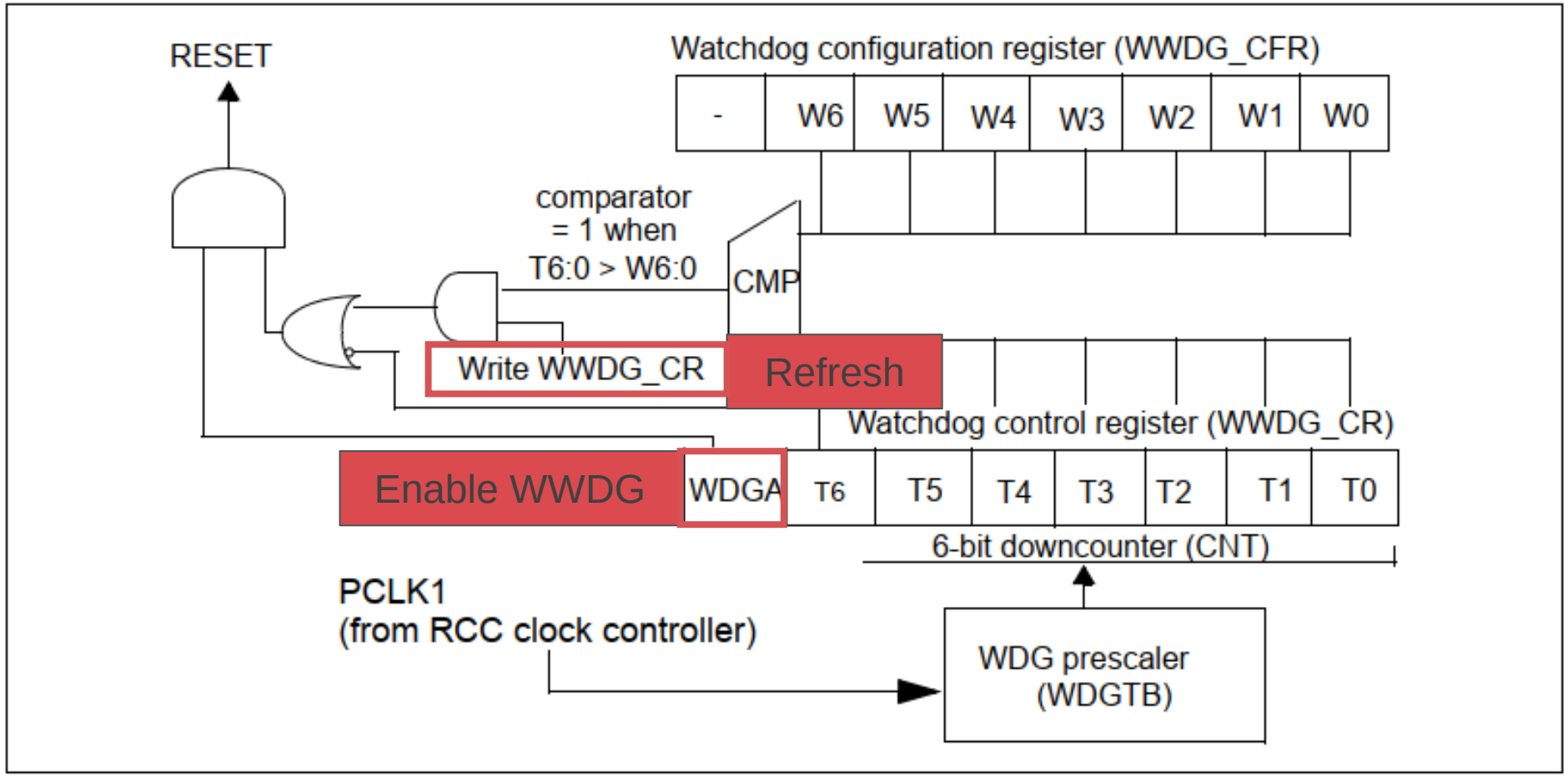
-- WWDG functional description

- If the watchdog is activated (the WDGA bit is set in WWDG_CR register) and when the 7-bit downcounter (T[6:0] bits) rolls over from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset.
- If the software reloads the counter while the counter is greater than the value stored in window register, then a reset is generated.
- The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent an MCU reset. This operation must occur only when the counter value is lower than the window register value. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0:
  - Enabling the watch
  - Controlling the downcounter

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

- Controlling the downcounter
  - This downcounter is free-running: It counts down even if the watchdog is disabled.
  - When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.
  - The T[5:0] bits contain the number of increments which represents the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value due to the unknown status of the prescaler when writing to the WWDG_CR register.
  - The Configuration register (WWDG_CFR) contains the high limit of the window: To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x3F.
  - Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

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

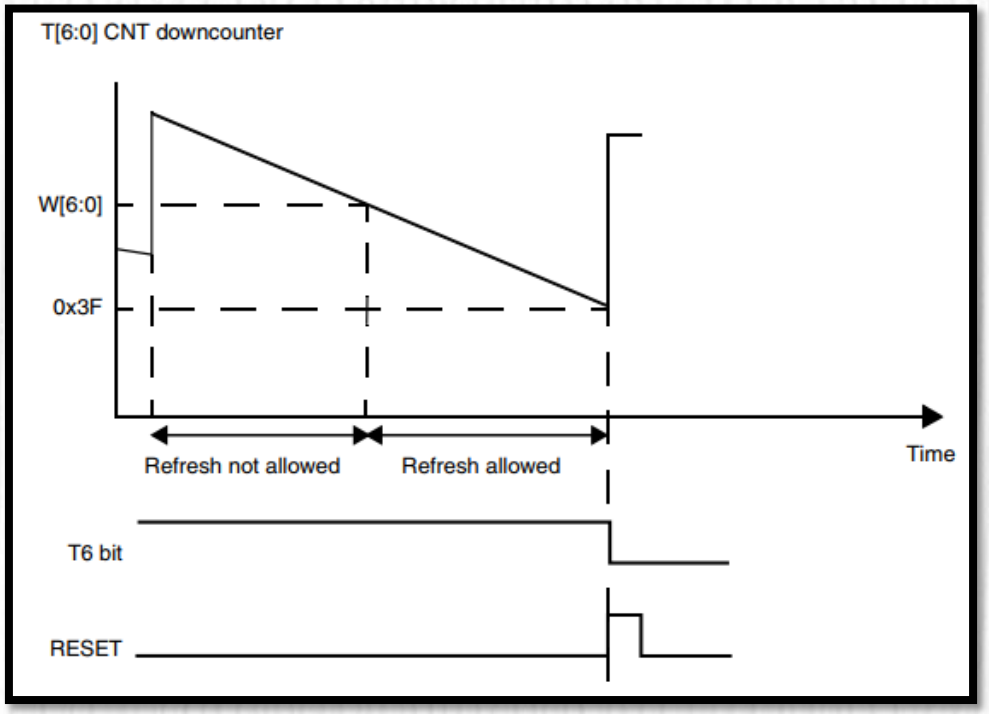


WWDG block diagram

1. WWDG Description

-- WWDG timeout calculation

- $T_{WWDG} = T_{pclk1} * 4096 * 2^{WDGTB} * (T[5:0] + 1)$
  - T_{pclk1} : APB1clock period



Window watchdog timing diagram

Min-max timeout value @36 MHz

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

- Enable watchdog clock
- Set frequency division coefficient
- Set upper window value
- Enable early wakeup interrupt (EWI) and group (optional)
- Enable watchdog
- Refresh the value of watchdog t [6:0] (feed dog)
- Write interrupt service function

02

WWDG Registers

2. WWDG Registers

- WWDG_CR: Control register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								WDGA	T[6:0]						
								rs	rw						

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **WDGA:** Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]:** 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter. It is decremented every (4096 x 2^{WDGTB}) PCLK1 cycles. A reset is produced when it rolls over from 0x40 to 0x3F (T6 becomes cleared).

2. WWDG Registers

- WWDG_CFR: Configuration register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						EWI	WDGTB[1:0]		W[6:0]						
						rs	rw		rw						

Bit 31:10 Reserved, must be kept at reset value.

Bit 9 **EWI:** Early wakeup interrupt

When set, an interrupt occurs whenever the counter reaches the value 0x40. This interrupt is only cleared by hardware after a reset.

Bits 8:7 **WDGTB[1:0]:** Timer base

The time base of the prescaler can be modified as follows:

00: CK Counter Clock (PCLK1 div 4096) div 1
01: CK Counter Clock (PCLK1 div 4096) div 2
10: CK Counter Clock (PCLK1 div 4096) div 4
11: CK Counter Clock (PCLK1 div 4096) div 8

Bits 6:0 **W[6:0]:** 7-bit window value

These bits contain the window value to be compared to the downcounter.

2. WWDG Registers

- WWDG_SR: Status register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															EWIF
															rc_w0

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EWIF:** Early wakeup interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing ‘0. A write of ‘1 has no effect. This bit is also set if the interrupt is not enabled.

03

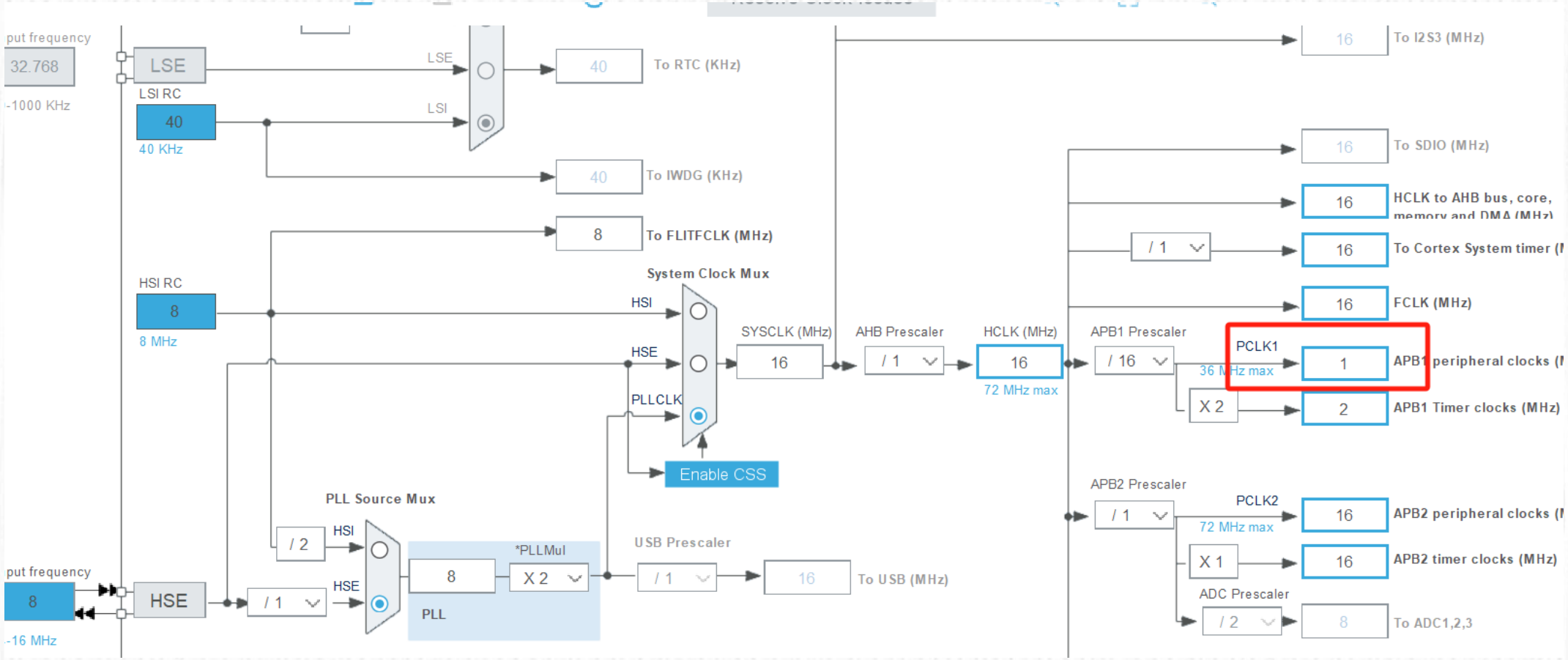
How to Program

3. How to Program

- Our Goal
  - Use EWI to make LED1 light on.
  - 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 period.

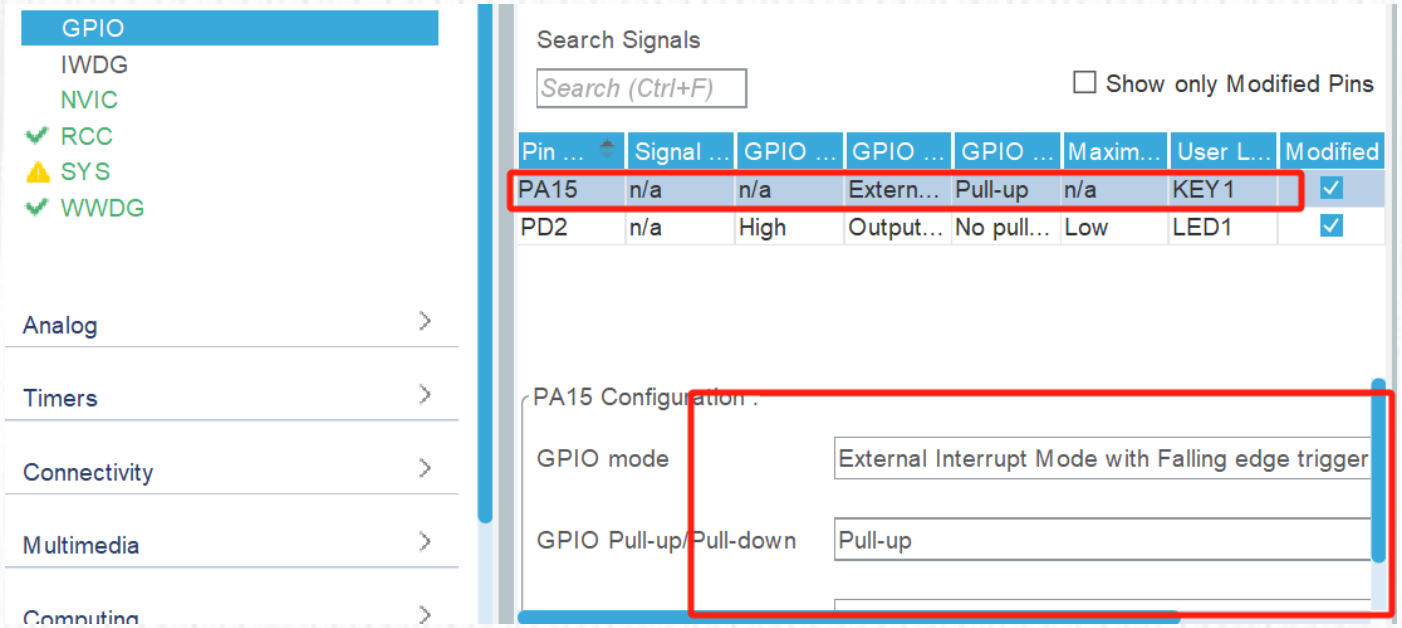
3. How to Program

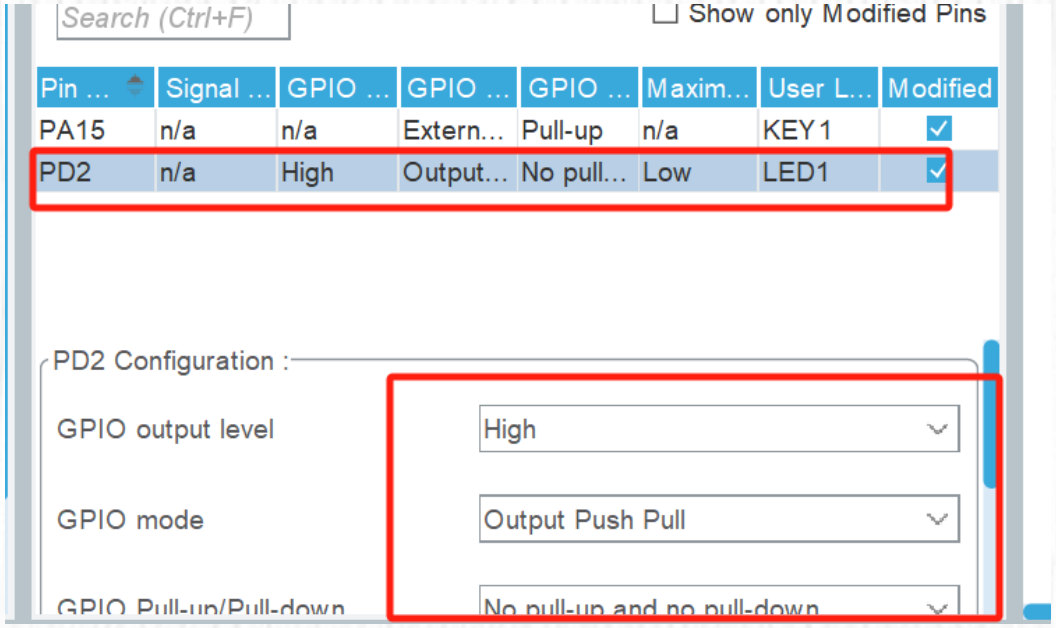
- Configure RCC



3. How to Program

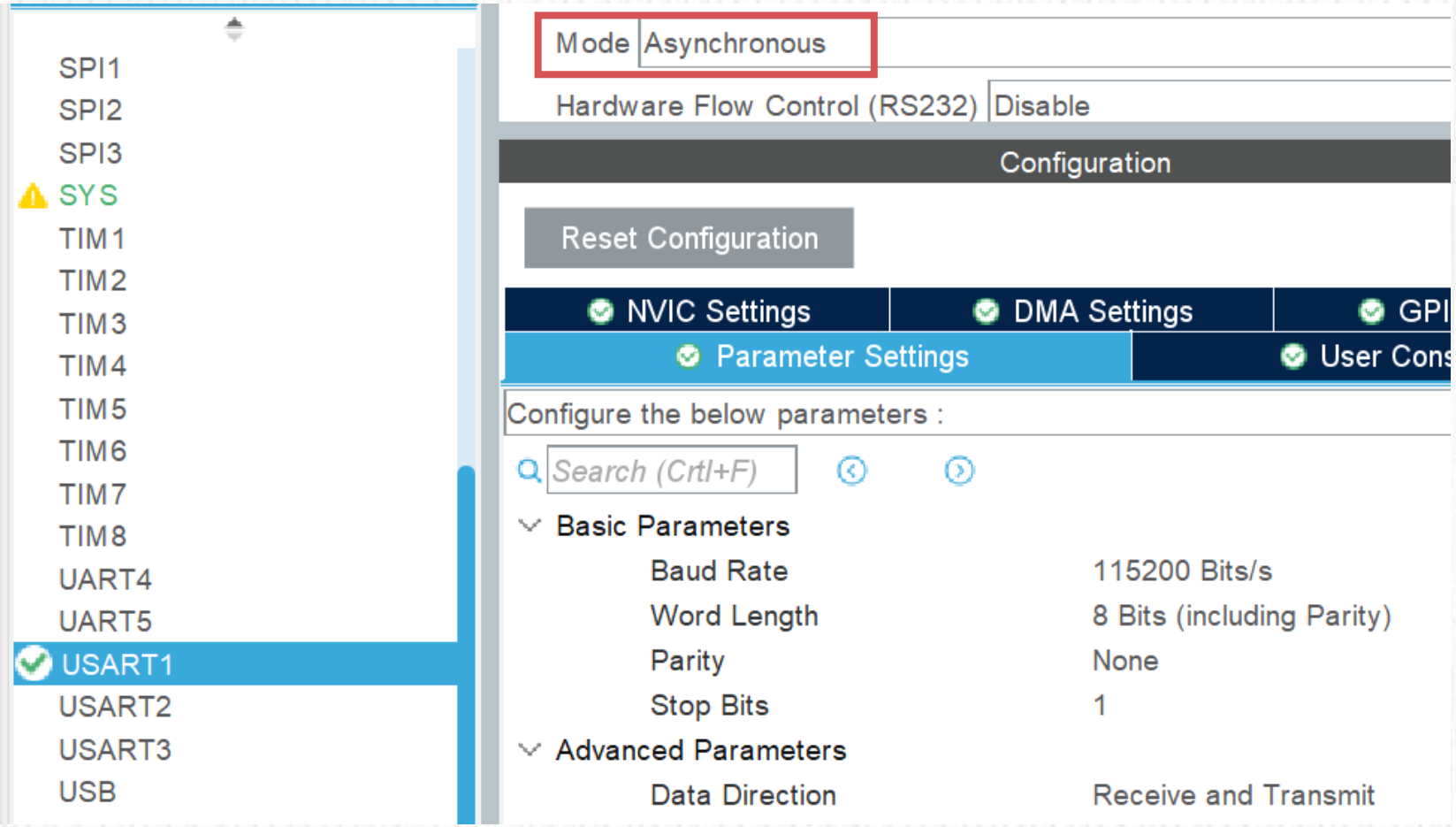
- Configure GPIO
  - Set PA15(KEY1) as external interrupt source
  - Set PD2(LED1) as output push-pull





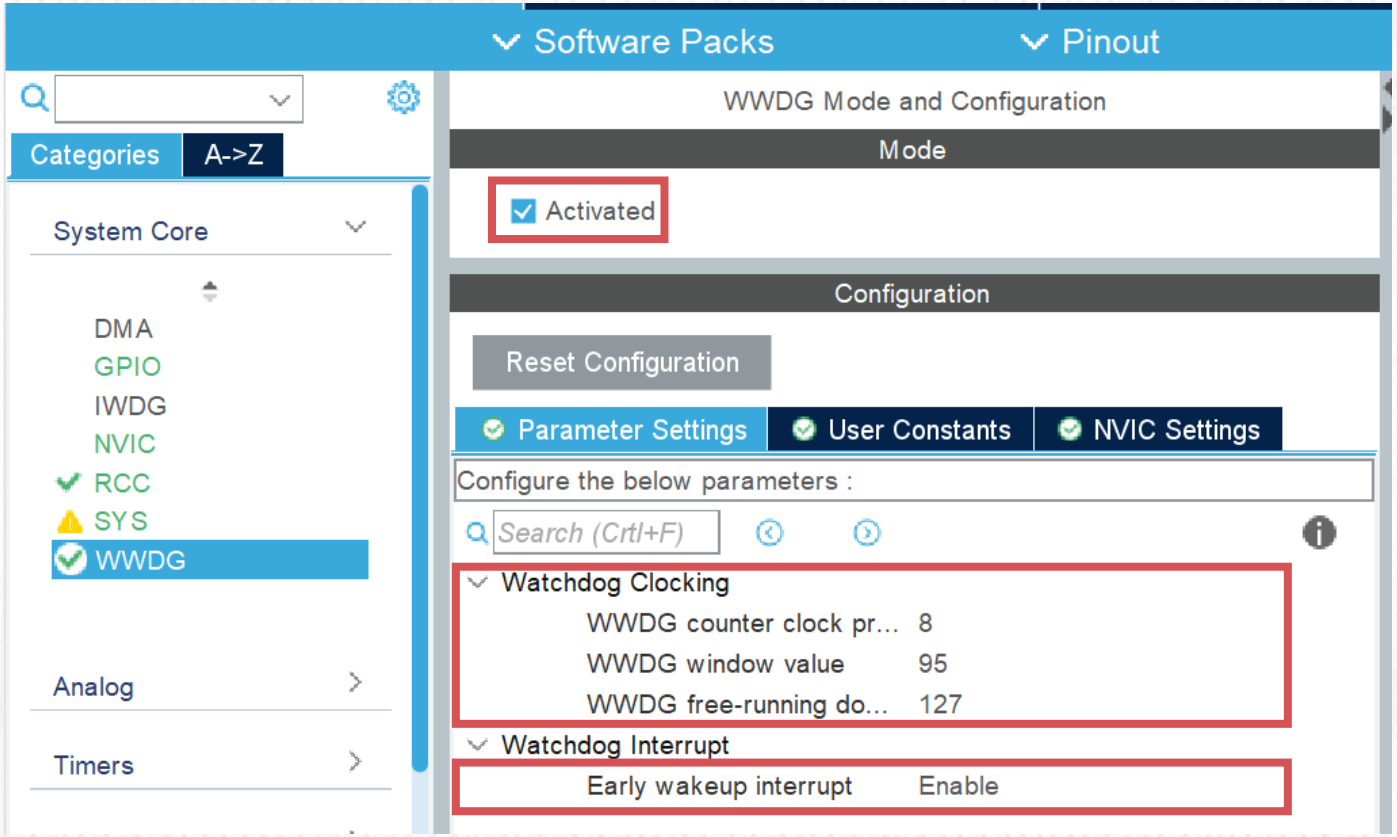
3. How to Program

- Configure USART1
  - Set the USART1 as asynchronous mode



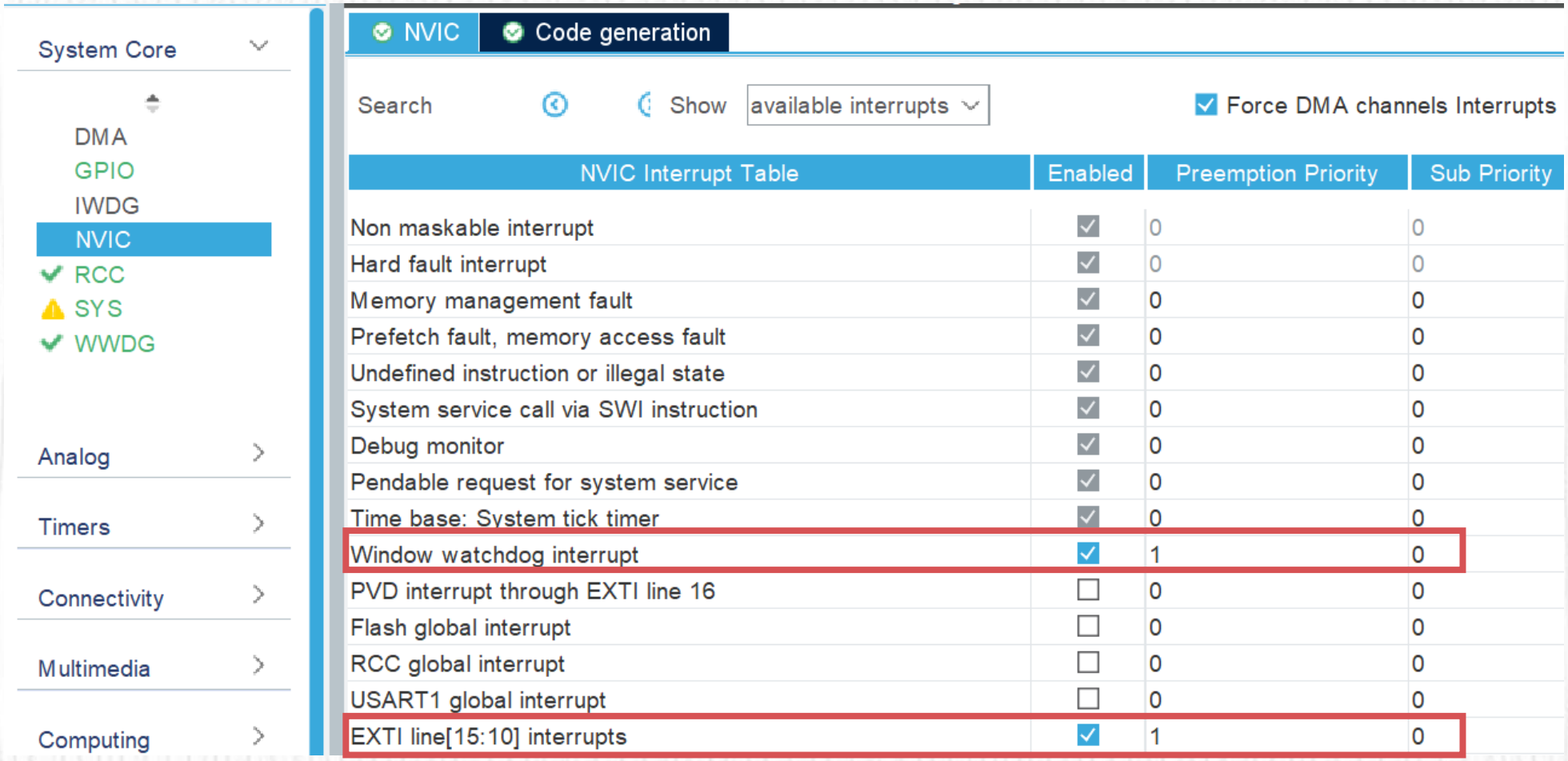
3. How to Program

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



3. How to Program

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



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 light up the Green LED(PD2) in EWI callback function, to show critical information: when EWI is triggered, the system has software fault already.
- Let the interrupt finish in 1 cycle, because when the counter is 0x3F, MCU will be reset.

```
void HAL_WWDG_EarlyWakeupCallback(WWDG_HandleTypeDef* hwwdg)
{
        HAL_GPIO_WritePin(GPIOD, LED1_Pin, RESET);
}
```

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.

```
HAL_GPIO_WritePin(GPIOD, LED1_Pin, SET);
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. Demo result

- The program is periodically reset, the value of variable i comes back to 1 (shown as Fig.1).The Green LED blinks.
- While pushing button KEY1, the program still can not run continuously(shown as Fig.2).

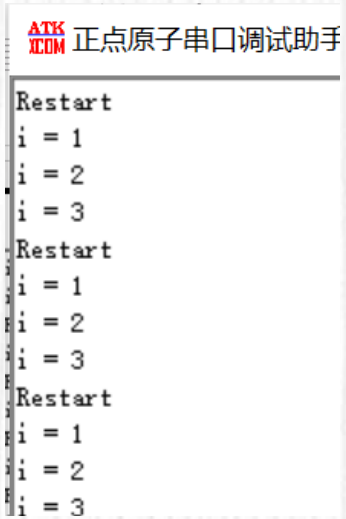
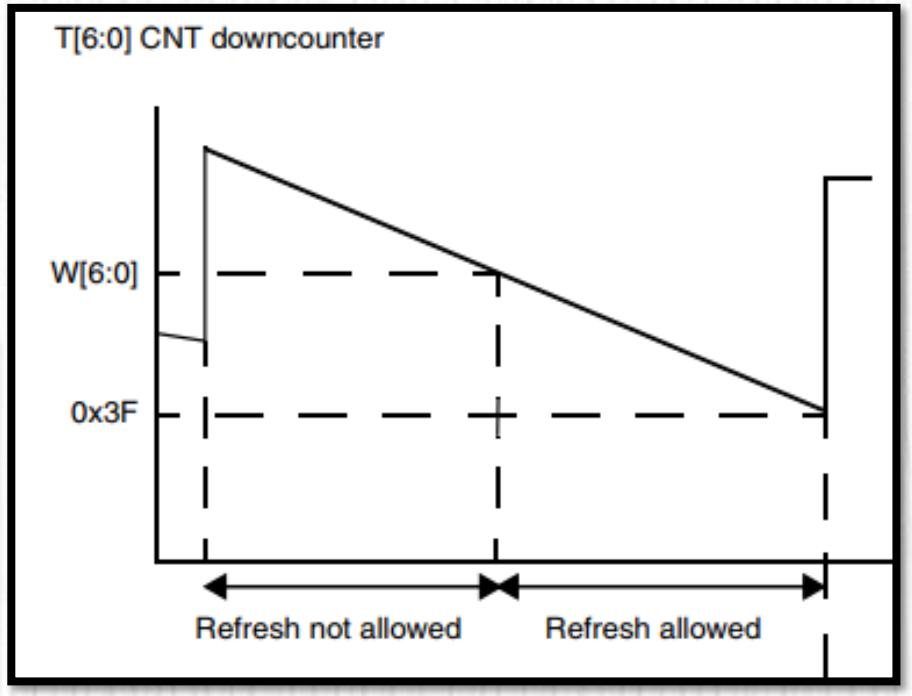


Fig.1 By default: The program resets periodically, Green LED blinks

Fig.2 Pushing button can not correctly feed dog



04

Practice

4. Practice

- Run the WWDG demo on MiniSTM32 board.
- Explain the reason why pushing button KEY1, the program still can not run continuously.
- Tell the time range in the demo to refresh WWDG without resetting it.
- Disable the EWI and EXTI, use a timer interrupt to refresh WWDG in a suitable time to avoid resetting CMU, and tell the parameters of WWDG and TIMER of your design.

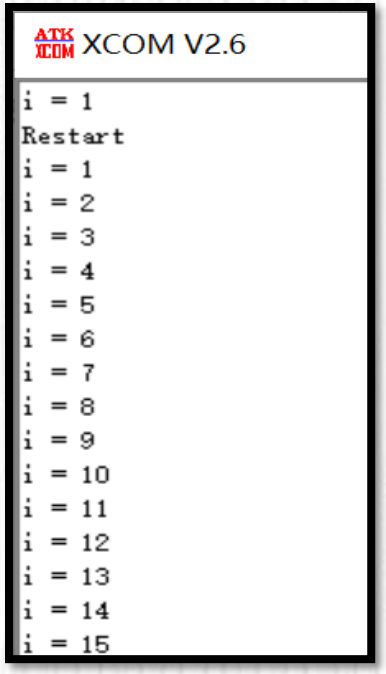


Fig.3 The program runs continuously

- TIPS: When using timer interrupt, you should clear the interrupt flag before HAL_TIM_Base_Start_IT(&htim3) function.

```
__HAL_TIM_CLEAR_IT(&htim3,TIM_IT_UPDATE);
```