Systick and Watchdog timer

Systick overview

- 24-bit count-down counter
- Can generate interrupt
- Systic source could be any of the following:
  - FCLK
  - External STCLK
- Widely use to support real time system whereas TIMx mainly used for peripheral

Systick system

- Control and status register
- Reload register (for reload with new value)
- Systick counter
- Timer adjust register

Example of code

```
unsigned long ms = 0;
int main(){
  RCC_setup();
  SYSTICK_setup();
  ms = 0;
  if(SysTick_GetFlagStatus(SysTick_FLAG_COUNT)==RESET){
        Systick_CounterCmd(SysTick_Counter_Enable);
 }

  while(1);
}
```

```
void RCC_setup(){
  ErrorStatus HSEStartUpStatus; // Keep error status
  RCC_DeInit();      // RCC system reset(for debug purpose)
  RCC_HSEConfig(RCC_HSE_ON);  // Enable HSE
  HSEStartUpStatus = RCC_WaitForHSEStartUp();   // Wait till HSE is ready
  if(HSEStartUpStatus == SUCCESS)
  {
    RCC_HCLKConfig(RCC_SYSCLK_Div1); // HCLK = SYSCLK
    RCC_PCLK2Config(RCC_HCLK_Div1); // PCLK2 = HCLK
    RCC_PCLK1Config(RCC_HCLK_Div2); // PCLK1 = HCLK/2
    RCC_ADCCLKConfig(RCC_PCLK2_Div4);  // ADCCLK = PCLK2/4
    FLASH_SetLatency(FLASH_Latency_2); // Flash 2 wait state
    FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);
    RCC_PLLConfig(RCC_PLLSource_HSE_Div1,RCC_PLLMul_9);
    RCC_PLLCmd(ENABLE);  // Enable PLL
    while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET);

    RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);  // Select PLL as system
    clock source
    while(RCC_GetSYSCLKSource() != 0x08); // Wait till PLL is used as system clock
    source
  }
```

```
void SysTick_setup(){
  SysTick_SetReload(9000);  // set reload value = 9000 (1ms)
   // fclk = hclk /8 = 9MHZ (default)
  SysTick_ITConfig(ENABLE); // Enable interrupt
}

void SysTickHandler(void){
   ms++;  // increment counter;
}
```

Watchdog timer

- STM32F10xxx supports two watchdog timers called independent timer and windows watchdog timer
- The watchdog timers are used for high safety level, and high timing accuracy
- The watchdog will trigger system reset or interrupt (window watchdog only) when the counter reaches a given timeout value

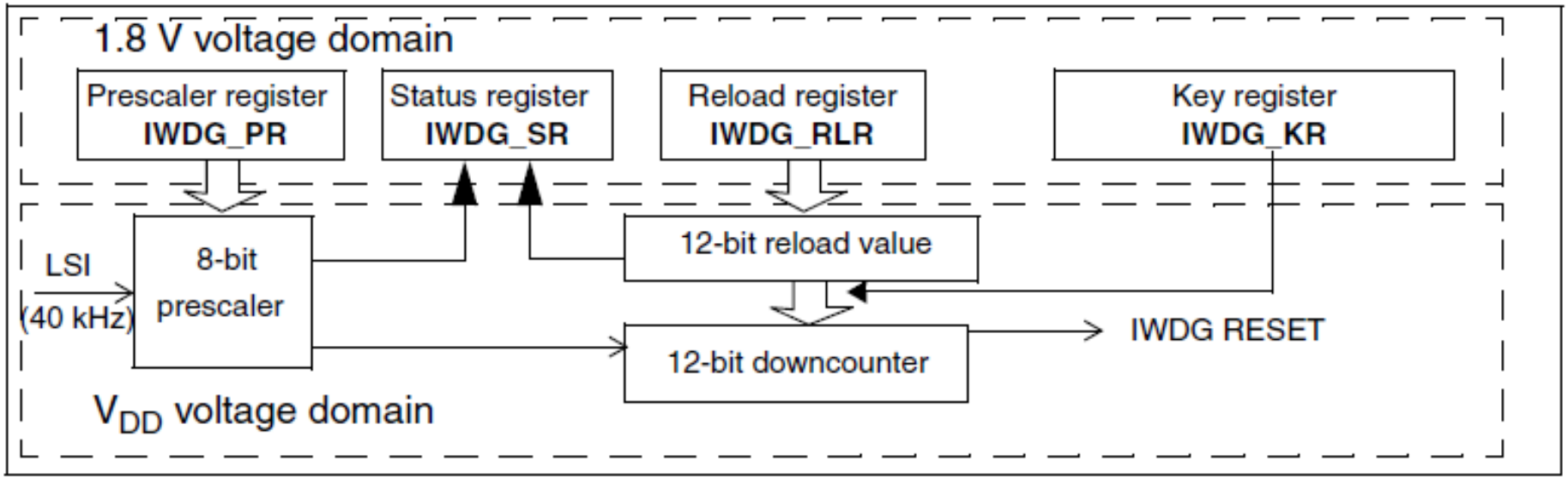
Independent Watchdog Timer (IWDG)

- IWDG is clocked by low-speed clock (LSI)
- It provides free running down counter
- It can operate in standby or stop modes
- Reset when the downcounter values of 0x00 is reached
- The watchdog is still functional in stop and standby mode

Hardware Components

- Prescaler register
- Status register
- Reload register
- Key register

IWDG block diagram



IWDG function

- When the key register is set to 0xCCCC, the watchdog will count down from the reset value of 0xFFF
- When the counter reaches 0x000, a reset signal is generated
- When the key register is set to 0xAAAA, the IWDG_RLR is reloaded to the counter, and the reset is prevented

Min and Max timeout period

- Min-time out is only 1 count

Table 94. Min/max IWDG timeout period at 40 kHz (LSI) (1)

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

Example

```
int main(){
  RCC_setup();
  NVIC_setup();
  GPIO_setup();
if(RCC_GetFlagStatus(RCC_FLAG_IWDGRST)!=RESET){ // check resume
     from IWDG reset?
     RCC_ClearFlag();  // clear reset flag
   }
   EXTI_setup();
   SYSTICK_setup();
    IWDG_setup();
   while(1);
}
```

```
void GPIO_setup(){
  GPIO_InitTypeDef GPIO_InitStructure;
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB|RCC_APB2Periph_GPIOC
     |RCC_APB2Periph_AFIO,ENABLE);
   GPIO_InitStructure.GPIO_Pin   =  GPIO_Pin_9;
   GPIO_InitStructure.GPIO_Mode  =  GPIO_Mode_IN_FLOATING;
   GPIO_Init(GPIOB, &GPIO_InitStructure);
   GPIO_InitStructure.GPIO_Pin   =  GPIO_Pin_6;
   GPIO_InitStructure.GPIO_Speed =  GPIO_Speed_50MHz;
   GPIO_InitStructure.GPIO_Mode  =  GPIO_Mode_OUT_PP;
   GPIO_Init(GPIOC, &GPIO_InitStructure);
}
void NVIC_setup(){
 NVIC_InitTypeDef NVIC-InitStructure;
  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);  // 2 bit for preemption priority
  NVIC_InitStructure.NVIC_IRQChannel = EXTI9_5_IRQChannel; //Enable EXT9 Interrup
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
  NVIC_SystemHandlerPriorityConfig(SystemHandler_Systick, 1, 0); // set prioirty Systick
     to 1
}
```

```
 void EXTI_setup()[
  EXTI_InitTypeDef EXTI_InitStructure;
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE);
  // Configure EXT1 line 9 to generate an interrupt on falling edge
    GPIO_EXTLineConfig(GPIO_PortSourceGPIOB, GPIO_PinSource0);
    EXTI_InitStructure.EXTI_Line = EXTI_Line9;
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;
    EXTI_Init(&EXTI_InitStructure);
}
void IWDG_setup(){
 IWDG_WriteAccessCmd(IwDG_WriteAccess_Enable);
 IWDG_SetPrescaler(IWDG_Prescaler_32); // set clock to 32 KHz/32 = 1 KHz
 IWDG_SetReload(349); //  reload every 350 ms
 IWDG_ReloadCounter();
 IWDG_Enable();
}
void SYSTICK_setup(){
  SysTick_SetReload(9000);  // set reload interval = 1ms
  SysTick_ITConfig(ENABLE);  // enable systick interrupt
  SystTick_CounterCmd(SystTick_Counter_Enable);  // enable systick counter
}
```

```
void SysTickHandler(){
  IWDG_ReloadCounter();  // reload IWDG counter
  ms++;
}

void EXTI9_5_IRQHandler(){
   if(EXTI_GetITStatus(EXTI_Line9) != RESET){
        GPIO_WriteBit(GPIOC,GPIO_Pin_6,1);  // turn on lled
        // EXTI line 9 pending bit is not clear, CPU will execute in this ISR indefinitely
        // Hence, does not service SysTick interrupt and will go to IDWG reset
   }
}
```

What will happen with this program?

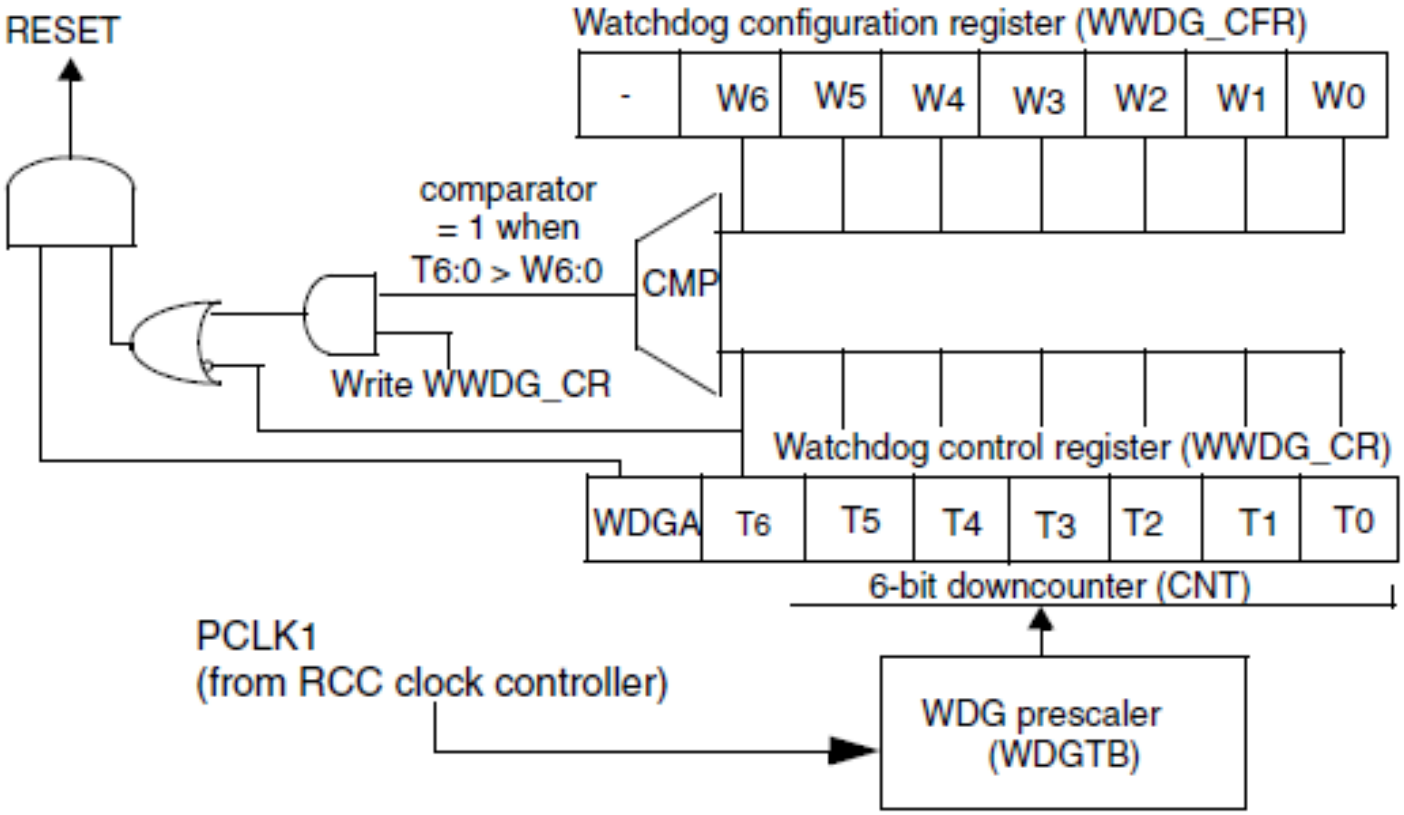
Solution

- Systick timer will reload every 1 ms
- Watchdog will count down every 350 ms
- After pressing the switch, interrupt flag has never been clear

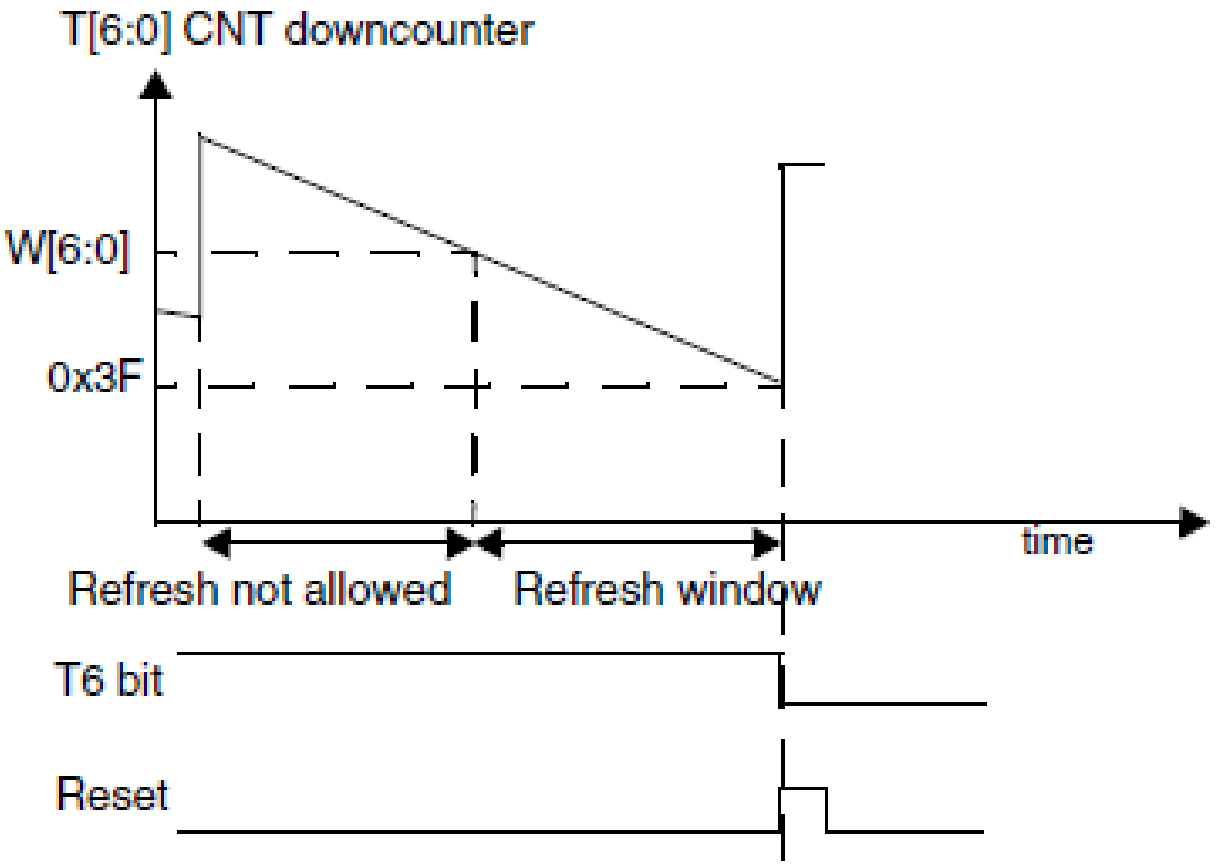
Window watchdog (WWDG)

- Use to detect software fault
- The watchdog generates the MCU reset on expiry of the counter, unless the program refresh the content of the counter
- The MCU reset is also generated, if the refresh happens before the window register value
- This implies that the counter must be refreshed during a limited window

Hardware diagram



Watchdog timing diagram



Data reload

- To prevent from the reset, WWDG provides WWDG_CFR register that will interrupt before watchdog expires
- The interrupt name is EWI, so that the CPU knows that WWDG will soon reset the system
- To prevent from reset, interrupt code should reload the value of WWDG counter

Min-Max time

Min-max timeout value @36 MHz (PCLK1)

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

Example

```
tnt wwdg_count=0;
int main(){
  RCC_setup();
  NVIC_setup();
  GPIO_setup();
if(RCC_GetFlagStatus(RCC_FLAG_WWDGRST)!=RESET){ // check resume
     from WWDG reset?
     RCC_ClearFlag();  // clear reset flag
   }
   EXTI_setup();
   WWDG_setup();
   while(1);
}
```

```
void GPIO_setup(){
  GPIO_InitTypeDef GPIO_InitStructure;
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB|RCC_APB2Periph_GPIOC
     |RCC_APB2Periph_AFIO,ENABLE);
   GPIO_InitStructure.GPIO_Pin   =  GPIO_Pin_9;
   GPIO_InitStructure.GPIO_Mode  =  GPIO_Mode_IN_FLOATING;
   GPIO_Init(GPIOB, &GPIO_InitStructure);
   GPIO_InitStructure.GPIO_Pin   =  GPIO_Pin_6;
   GPIO_InitStructure.GPIO_Speed =  GPIO_Speed_50MHz;
   GPIO_InitStructure.GPIO_Mode  =  GPIO_Mode_OUT_PP;
   GPIO_Init(GPIOC, &GPIO_InitStructure);}
void NVIC_setup(){
 NVIC_InitTypeDef NVIC-InitStructure;
  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);  // 2 bit for preemption priority
  NVIC_InitStructure.NVIC_IRQChannel = EXTI9_5_IRQChannel; //Enable EXT9 Interrup
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
  NVIC_InitStructure.NVIC_IRQChannel = WWDG_IRQChannel;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;
NVIC_Init(&NVIC_InitStructure);
}
```

```
 void EXTI_setup()[
  EXTI_InitTypeDef EXTI_InitStructure;
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE);
  // Configure EXT1 line 9 to generate an interrupt on falling edge
    GPIO_EXTLineConfig(GPIO_PortSourceGPIOB, GPIO_PinSource0);
    EXTI_InitStructure.EXTI_Line = EXTI_Line9;
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;
    EXTI_Init(&EXTI_InitStructure);
}

void IWDG_setup(){
RCC_APB2PeriphClockCmd(RCC_APB1Periph_WWDG,ENABLE);
WWDG_SetPrescaler(WWDG_Prescaler_8); // set to PCLK1 KHz/8 = 1098Hz = 0.91ms
 WWDG_SetWindowValue(0x41); //  set interrupt = 65
 WWDG_Enable(0x7f);   // reset counter
 WWDG_ClearFlag();  // clear EWIflag
  WWDG_EnableT();  // enalbe interrupt
}
```

```
void WWDG_IRQHandler(){
  WWDG_SetCounter(0x7f);  // reload WWDG counter
  WWDG_ClearFlag();  //  clear EWI  interrupt flag
   wwdg_count++;
}

void EXTI9_5_IRQHandler(){
   if(EXTI_GetITStatus(EXTI_Line9) != RESET){
        GPIO_WriteBit(GPIOC,GPIO_Pin_6,1);  // turn on lled
        // EXTI line 9 pending bit is not clear, CPU will execute in this ISR indefinitely
        // Hence, does not service WWDG_IRQ interrupt and will go to IDWG reset
   }
}
```

What will happen with this program?

Solution

- Interrupt will reload every 65x0.91 ms
- Watchdog will reset every 113x0.91 ms
- After pressing the switch, interrupt flag has never been clear

Question?

- Write the program to generate RTC interrupt every 50 ms to clear Window Watchdog timer of 100x0.91 ms

Questions?