

5. タイマー割り込み

[STM32F4DISCOVERY/STM32CubeIDE で、freeRTOS を使う]

2020年8月29日

(骨子)

TIM1 を使う。

プリスケーラ × カウンタピリオド × 1/f = タイマー周期

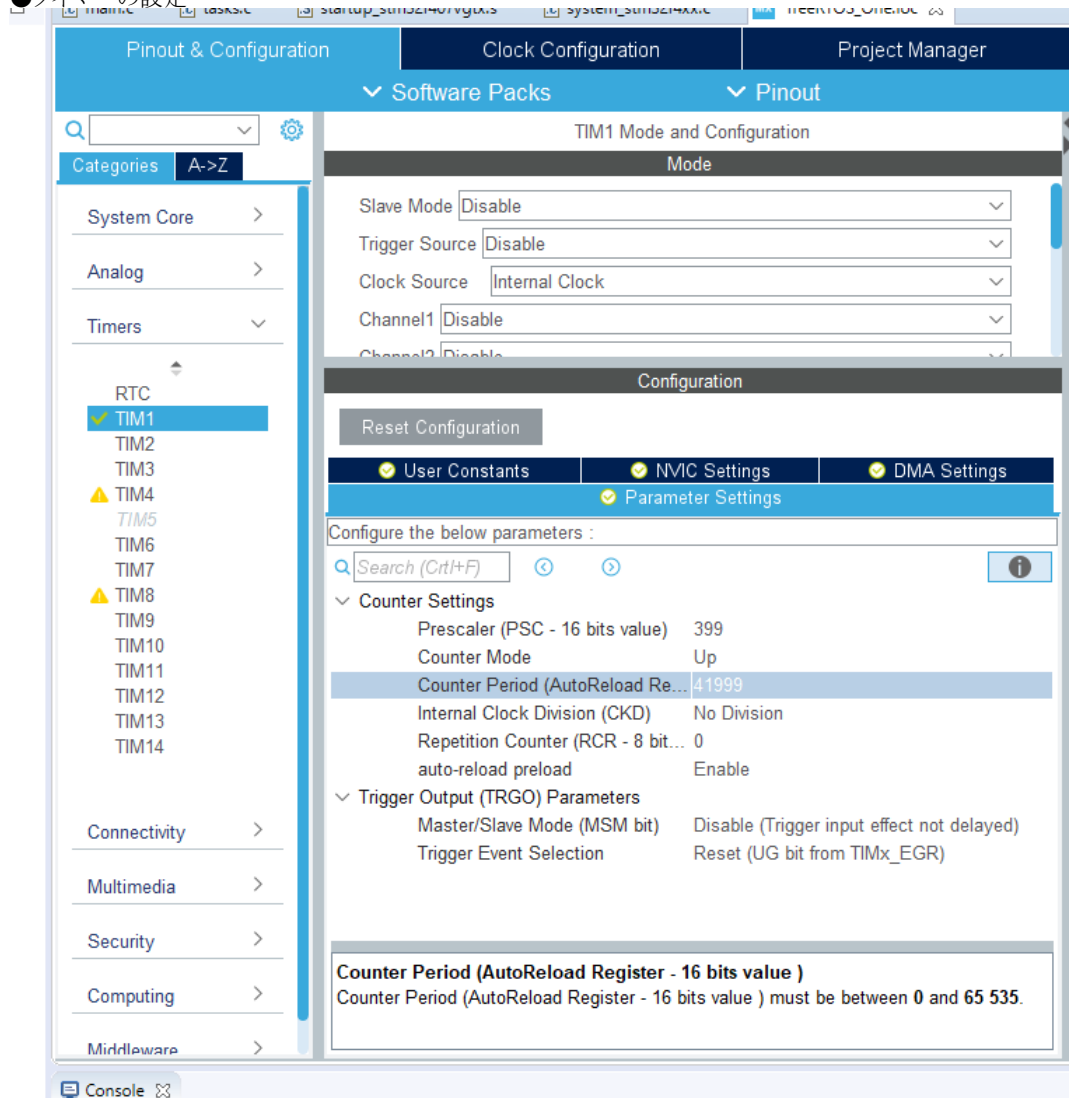
$400 \times 42000 \times 1/168M = 100ms$

クロック周波数 f = 168MHz(後述参照)

APB1 : TIM2, 3, 4, 5

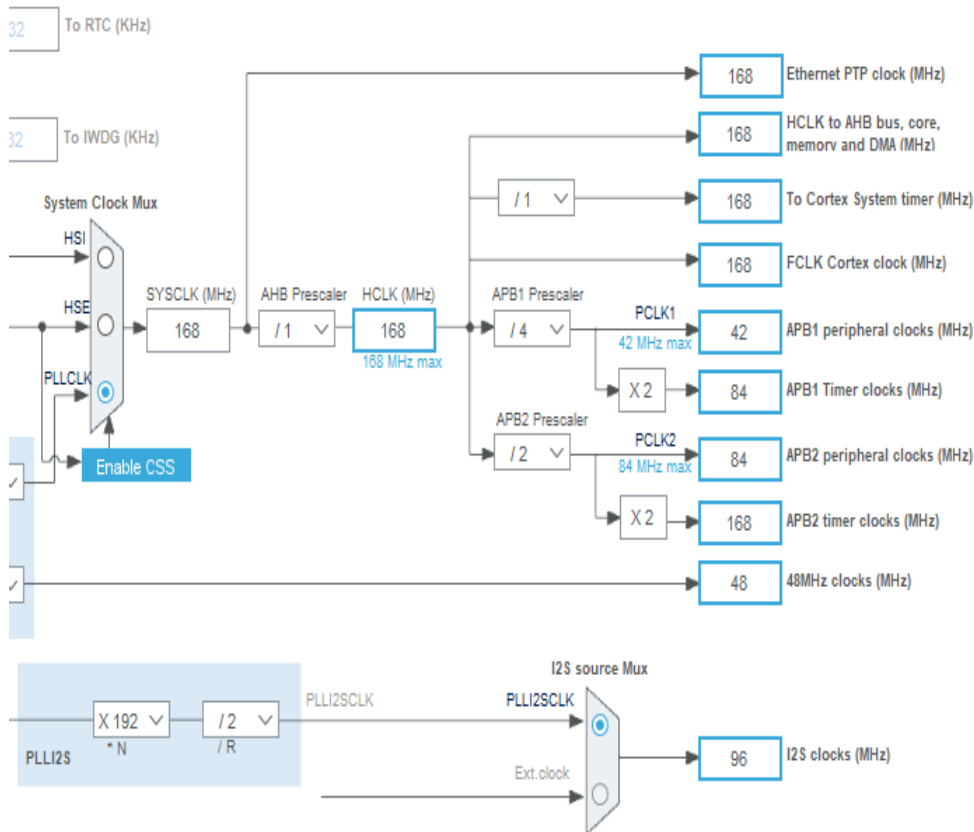
APB2 : TIM1, 9, 10, 11

●タイマーの設定



※Clock Source = Internal Clock。 Prescaler、Counter Period には1小さい値を設定する。

●クロック周波数の確認



※APB2=168MHz --> TIM1

●割り込みを発生させる

NVIC

✓ RCC

▲ SYS

WWDG

Analog >

Timers v

RTC

✓ TIM1

TIM2

TIM3

TIM4

Channel2 | Disable

Configuration

Reset Configuration

✓ NVIC Settings

✓ DMA Settings

✓ Parameter Settings

✓ User Constants

NVIC Interrupt Table	Ena...	Preemption ...	Sub Pr...
TIM1 break interrupt and TIM9 global interrupt	<input type="checkbox"/>	0	0
TIM1 update interrupt and TIM10 global interrupt	<input checked="" type="checkbox"/>	5	0
TIM1 trigger and commutation interrupts and ...	<input type="checkbox"/>	0	0
TIM1 capture compare interrupt	<input type="checkbox"/>	0	0

■実装(100ms 周期のタイマー割り込みの中からシグナルイベントを発生させ、タスクで受けて LED をトグルする)

●初期化(要コーディング)

```
HAL_TIM_Base_Start_IT(&htim1);
```

●割り込みハンドラ(TIM 共通)

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    /* USER CODE BEGIN Callback 0 */
    /* USER CODE END Callback 0 */

    /* USER CODE BEGIN Callback 1 */
    if (htim == &htim1)
    {
        osSignalSet(defaultTaskHandle, (1<<1));
    }
    /* USER CODE END Callback 1 */
}
```

■割り込みハンドラが本当に割り込みの中で呼ばれているか確認する。

●割り込みハンドラにブレークをかけて、xpsr の値を調べる。

※xpsr の下位 8 ビットが 0 以外なので、割り込みハンドラであることが分かる。

※割り込みハンドラの中から、シグナルイベントを発生させている。

The screenshot shows an IDE with the following components:

- Source Code Editor:** Displays the `HAL_TIM_PeriodElapsedCallback` function. Line 533, `osSignalSet(defaultTaskHandle, (1<<1));`, is highlighted with a blue selection bar.
- Register Window:** Located on the right, it lists various registers. The `xpsr` register is highlighted in blue, showing a value of `0x61000029 (Hex)`. Other registers like `r8` through `r12`, `sp`, `lr`, `pc`, `d0` through `d3` are also visible.
- Variable Information Window:** Below the register window, it shows details for the `xpsr` variable: Name: xpsr, Hex: 0x61000029, Decimal: 1627389993, Octal: 014100000051, Binary: 110000100000000000000000, Default: 1627389993.
- Bottom Panel:** Contains tabs for Console, Problems, Executables, Debugger Console, and Memory.

- シグナルイベントを受けるタスク側にブレークをかけて、xpsr の値を調べる。
※xpsr の下位 8 ビットが 0 なので、タスク(割り込みハンドラではない)ことが分かる。
※シグナルイベントを受けて LED の点灯をトグルしている。

The screenshot shows the STM32CubeIDE interface. On the left, the 'main.c' file is open, displaying the `StartDefaultTask` function. The function is an infinite loop that waits for a signal event. When a signal is received, it toggles two LEDs (LD6 and LD4). The line `HAL_GPIO_TogglePin(LD4_GPIO_Port, LD4_Pin);` is highlighted with a blue selection bar. On the right, the 'Registers' window is open, showing the values of various registers. The `xpsr` register is highlighted in blue, with a value of `0x21000000 (Hex)`. Below the register list, a detailed view of the `xpsr` register is shown, including its hexadecimal, decimal, octal, and binary representations.

Name	Value	De
r8	0	
r9	0	
r10	0	
r11	0	
r12	0	
sp	0x20000620 <uc...	
lr	134258235	
pc	0x8000a90 <Start...	
xpsr	0x21000000 (Hex)	
d0	0	
d1	0	
d2	0	
d3	0	

Name : xpsr
 Hex: 0x21000000
 Decimal: 553648128
 Octal: 04100000000
 Binary: 100001000000000000000000
 Default: 553648128

■参考URL

STM32CubeIDE で FreeRTOS を使ってみる

<https://moons.link/post-117/>

(コンフィグレーターを使ってみる タイマー割り込み編)

<https://moons.link/post-50/>