TIM4 Output Channels - Output Compare Mode

ဒီတစ်ခါတော့ TIM4 နဲ့ Output Compare Mode လေးကိုစမ်းပြချင်ပါတယ်။ ပထမဦးဆုံး TIM4 အတွက် clock ပေးလိုက်ရအောင်။

6.3.13 RCC APB1 peripheral clock enable register (RCC_APB1ENR)

Address offset: 0x40

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 EN	UART7 EN	DAC EN	PWR EN	Reser-ved	CAN2 EN	CAN1 EN	Reser-ved	I2C3 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART 3 EN	USART 2 EN	Reser-ved
rw	rw	rw	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	Reserved		WWDG EN	Reserved		TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

RCC_APB1ENR ကိုသွားကြည့်လိုက်တော့ TIM4 ဟာ Bit 2 မှာရှိတာကိုတွေ့ရပါတယ်။ အဲဒီအတွက်ကြောင့် TIM4 ကို clock ပေးဖို့ရန်အတွက် အခုလိုမျိုးရေးလိုက်ပါတယ်။

```
__setbit(RCC->APB1ENR, 2U);
```

အပေါ်က "Timer Blinky" သင်ခန်းစာမှာပြောခဲ့တဲ့အတိုင်း counter mode သတ်မှတ်မယ်၊ ARR တန်ဖိုးထည့်မယ်။ TIM4 ကို start လုပ်ပါမယ်။

ပြီးသွားရင်တော့ TIM4 ရဲ့ channel လေးခုစလုံးကို Output Mode ကြေညာမှာဖြစ်ပါတယ်။ TIM4 ရဲ့ CCMR1 ကိုသွားကြည့်ပါမယ်။

18.4.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. Take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

အဲဒီအထဲက bit 0 နဲ့ 1၊ bit 8 နဲ့ 9 ကိုဖတ်ပါမယ်။

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, IC1 is mapped on TI1.

10: CC1 channel is configured as input, IC1 is mapped on TI2.

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

```
__clearbit(TIM4->CCMR1, 0U);

__clearbit(TIM4->CCMR1, 1U);



__clearbit(TIM4->CCMR1, 8U);

__clearbit(TIM4->CCMR1, 9U);
```

00 ဆိုရင် Output အနေနဲ့ကြေညာတာဖြစ်လို့ အကုန်လုံးကို clear လုပ်ပစ်လိုက်ပါတယ်။

နောက်တစ်ခါထပ်ကြည့်ရမှာက TIM4 ရဲ့ CCMR2 ဖြစ်ပါတယ်။

18.4.8 TIMx capture/compare mode register 2 (TIMx_CCMR2)

Address offset: 0x1C

Reset value: 0x0000

Refer to the above CCMR1 register description.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

အဲဒီဇယားကနေ bit 8 နဲ့ 9၊ bit 0 နဲ့ 1 ကိုထပ်ကြည့်ပါတယ်။

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if

Bits 1:0 **CC3S**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

```
__clearbit(TIM4->CCMR2, 0U);

__clearbit(TIM4->CCMR2, 1U);

__clearbit(TIM4->CCMR2, 8U);

 __clearbit(TIM4->CCMR2, 9U);
```

00 ပေးတာက Output အနေနဲ့ကြေညာတာဖြစ်လို့ အကုန်လုံးကို clear လုပ်ပစ်လိုက်ပါတယ်။

Compare လုပ်တဲ့အပေါ်မူတည်ပြီးတော့ OCxREF signal ကို toggle ဖြစ်စေဖို့ရန်အတွက် CCMR1 ကိုထပ်ကြည့်ပါ မယ်။

18.4.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. Take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 6:4 **OC1M**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

100: Force inactive level - OC1REF is forced low.

101: Force active level - OC1REF is forced high.

110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1

အဲဒီမှာ ပြောထားတာကိုဖတ်ကြည့်လိုက်တော့ 011 ဆိုရင် Toggle ဖြစ်မယ်လို့ရေးထားတာဖြစ်ပါတယ်။ ဒါကြောင့် bit 12 နဲ့ 13, bit 4 နဲ့ 5 ကိုအကုန်လုံး set လုပ်လိုက်ပါမယ်။ CCMR1 တင်မဟုတ်ပါဘူး။ CCMR2 က bit 12 နဲ့ 13, bit 4 နဲ့ 5 ကိုပါ set လုပ်မှာဖြစ်ပါတယ်။

```
__setbit(TIM4->CCMR1, 4U);

__setbit(TIM4->CCMR1, 5U);
__setbit(TIM4->CCMR1, 12U);
__setbit(TIM4->CCMR1, 13U);

__setbit(TIM4->CCMR2, 4U);

 __setbit(TIM4->CCMR2, 5U);

 __setbit(TIM4->CCMR2, 12U);
 __setbit(TIM4->CCMR2, 13U);
```

ပြီးရင်တော့ channel လေးခုစလုံးအတွက် compare trigger point တွေသတ်မှတ်ပါမယ်။

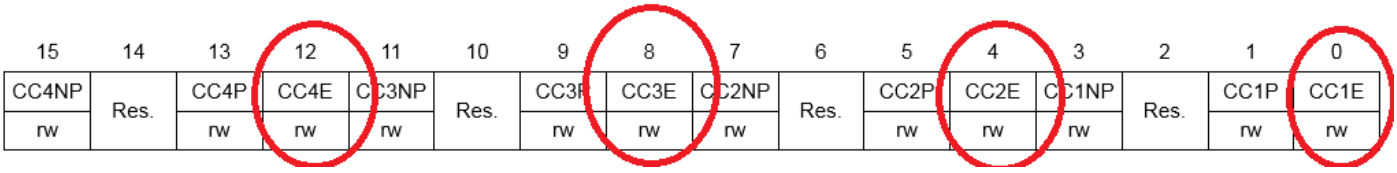
```
TIM4->CCR1 = 2000U;
TIM4->CCR2 = 4000U;
TIM4->CCR3 = 6000U;
TIM4->CCR4 = 8000U;
```

ပြီးရင်တော့ Output Compare Channel ကို enable လုပ်ပေးရမှာဖြစ်ပါတယ်။

18.4.9 TIMx capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

```
__setbit(TIM4->CCER, 0U);

__setbit(TIM4->CCER, 4U);

__setbit(TIM4->CCER, 8U);

__setbit(TIM4->CCER, 12U);
```

ပြီးရင် GPIOD အတွက် clock ပေးလိုက်ပါမယ်။ __setbit(RCC->AHB1ENR, 3);

ဒီသင်ခန်းမှာထုတ်မယ့် GPIO လေးခုက PD12, PD13, PD14, PD15 တို့ဖြစ်ပါတယ်။ အဲဒီ pin တွေကို alternate function အနေနဲ့သတ်မှတ်ခဲ့မှာဖြစ်ပါတယ်။

```
__clearbit(GPIOD->MODER, 24U);
__setbit(GPIOD->MODER, 25U);

__clearbit(GPIOD->MODER, 26U);

__setbit(GPIOD->MODER, 27U);

__clearbit(GPIOD->MODER, 28U);

  __setbit(GPIOD->MODER, 29U);

__clearbit(GPIOD->MODER, 30U);

 __setbit(GPIOD->MODER, 31U);
```

ယခုသင်ခန်းစာရဲ့ gpio လေး pin အတွက် alternate function သတ်မှတ်တဲ့နေရာမှာ ဘာလို့အပေါ်က အတိုင်း code ရေးရသလဲဆိုတော့ datasheet ကိုဖတ်ကြည့်လိုက်တဲ့အခါမှာ အောက်ပါအတိုင်း တွေ့ရလို့ဖြစ်ပါတယ်။

8.4.1 GPIO port mode register (GPIOx_MODER) (x = A..I/J/K)

Address offset: 0x00

Reset values:

- 0xA800 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **MODERy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O direction mode.
00: Input (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

Pin 12, 13, 14 နဲ့ 15 တို့ကို timer-4 ရဲ့ output compare channel နဲ့ချိတ်ဆက်မှာဖြစ်တဲ့အတွက် GPIOD ဟာ AF2(0x2) ကို implement လုပ်ရပါမယ်။

```
GPIOD->AFR[1] = 0x0U;
__setbit(GPIOD->AFR[1], 17U);
__setbit(GPIOD->AFR[1], 21U);
__setbit(GPIOD->AFR[1], 25U);
__setbit(GPIOD->AFR[1], 29U);
```

ဒီသင်ခန်းစာအတွက် ကုဒ်အပြည့်အစုံကိုပြပါတော့မယ်။

```
#include <stdint.h>

#include "stm32f4xx.h"

#define   __setbit(___reg, ___bit)     ((___reg) |= (1U << (___bit)))
#define   __clearbit(___reg, ___bit)   ((___reg) &= (~(1U << (___bit))))
#define   __togglebit(___reg, ___bit)  ((___reg) ^= (1U << (___bit)))
#define   __getbit(___reg, ___bit)     (((___reg) & (1U << (___bit))) >> (___bit))
```

```
static void initLEDs(void);
static void initTimer(void);
#define     UP_COUNTER      1

int main () {

    initLEDs();
    initTimer();
    for (;;) {}

}



static void initTimer(void) {
        __setbit(RCC->APB1ENR, 2U);
        TIM4->PSC = 1000;

        #if (UP_COUNTER)
                __clearbit(TIM4->CR1, 4U);
        #else
                __clearbit(TIM4->CR1, 4U);
        #endif

    TIM4->ARR = 8000;
__setbit(TIM4->CR1, 0U);

    __clearbit(TIM4->CCMR1, 0U);
        __clearbit(TIM4->CCMR1, 1U);
        __clearbit(TIM4->CCMR1, 8U);
        __clearbit(TIM4->CCMR1, 9U);

         __clearbit(TIM4->CCMR2, 0U);
        __clearbit(TIM4->CCMR2, 1U);
        __clearbit(TIM4->CCMR2, 8U);
        __clearbit(TIM4->CCMR2, 9U);
```

```
        __setbit(TIM4->CCMR1, 4U);
        __setbit(TIM4->CCMR1, 5U);
        __setbit(TIM4->CCMR1, 12U);
        __setbit(TIM4->CCMR1, 13U);

        __setbit(TIM4->CCMR2, 4U);
        __setbit(TIM4->CCMR2, 5U);
        __setbit(TIM4->CCMR2, 12U);
        __setbit(TIM4->CCMR2, 13U);

        TIM4->CCR1 = 2000U;
        TIM4->CCR2 = 4000U;
        TIM4->CCR3 = 6000U;
        TIM4->CCR4 = 8000U;

        __setbit(TIM4->CCER, 0U);
        __setbit(TIM4->CCER, 4U);
        __setbit(TIM4->CCER, 8U);
        __setbit(TIM4->CCER, 12U);
        }
static void initLEDs(void) {
        __setbit(RCC->AHB1ENR, 3);
    __clearbit(GPIOD->MODER, 24U);
    __setbit(GPIOD->MODER, 25U);
     __clearbit(GPIOD->MODER, 26U);
    __setbit(GPIOD->MODER, 27U);
    __clearbit(GPIOD->MODER, 28U);
    __setbit(GPIOD->MODER, 29U);
```

```
    __clearbit(GPIOD->MODER, 30U);

    __setbit(GPIOD->MODER, 31U);

    GPIOD->AFR[1] = 0x0U;

    __setbit(GPIOD->AFR[1], 17U);

    __setbit(GPIOD->AFR[1], 21U);

    __setbit(GPIOD->AFR[1], 25U);

    __setbit(GPIOD->AFR[1], 29U);

}
```