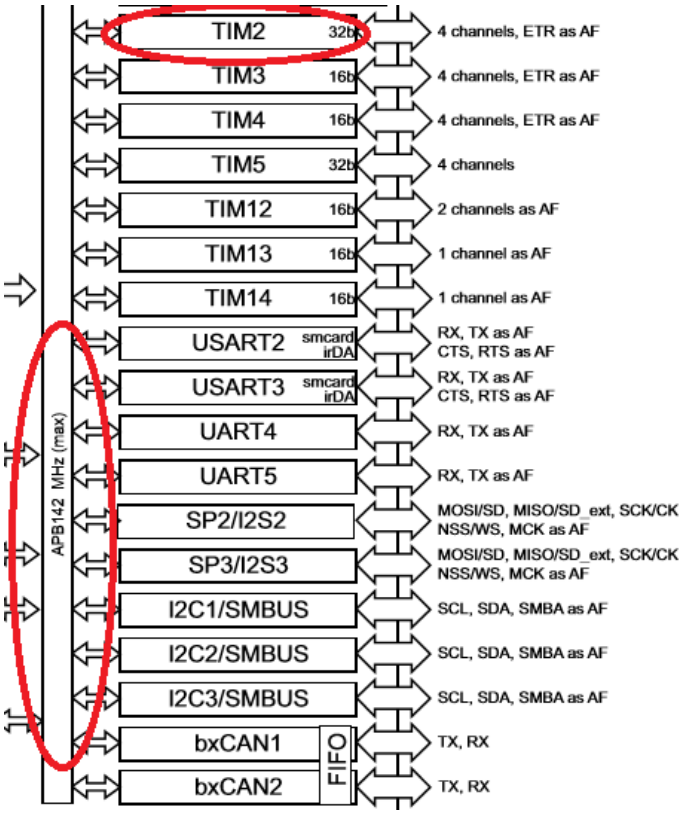
Bare-Metal Programming Timer Blink

ယခုတစ်ခါပြောမှာကတော့ Timer 2 ကိုသုံးပြီးတော့ Blink လုပ်တဲ့နည်းလမ်းကိုပြောမှာဖြစ်ပါတယ်။ Timer 2 ကိုသုံးမှာဖြစ်တဲ့အတွက် datasheet ကိုတစ်ချက်ကြည့်လိုက်ရအောင်။



အဲဒီ block diagram လေးမှာတွေ့ရတာက TIM2 ဟာ APB1 ကနေ clock ကိုရယူပါတယ်။ APB1 ဟာ 42 MHz ရှိပါတယ်။ APB1 ကိုသုံးမှာဖြစ်တဲ့အတွက်ကြောင့် Timer 2 ကို clock ပေးဖို့ရန်အတွက် datasheet ထဲက APB1ENR ဆိုတဲ့ enable register ဇယားကိုသွားကြည့်ပါမယ်။

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

TIM2 ဟာ bit 0 မှာရှိတယ်ဆိုတာကို အပေါ်ကဇယားလေးမှာမြင်ရမှာဖြစ်ပါတယ်။ အဲဒီအတွက် TIM2 အတွက် bit 0 ကို clock enable လုပ်ဖို့ရန် အောက်ပါ ကုဒ်ကိုရေးပါတယ်။

```
__setbit(RCC->APB1ENR, 0U);
```

Default configuration မှာ TIM2 က 16Mhz (HSI) bus clock ကိုယူပါတယ်။ အရင်ကရှင်းခဲ့တဲ့အတိုင်း PSC ကိုရေးပါမယ်။ Timer ရဲ့ input frequency ကိုစားလိုက်မှာဖြစ်ပါတယ်။

TIM2->PSC = 1000;

TIM2->PSC = 1000; ကိုရေးလိုက်တဲ့အတွက်ဘယ်လိုဖြစ်သွားသလဲဆိုတော့ -

16 MHz = 16,000,000/1,000 = 16,000 = 16Khz ဖြစ်သွားပါတယ်။

အခုစမ်းမှာက Up-Counter Mode နဲ့စမ်းမှာဖြစ်ပါတယ်။ ဒါ့ကြောင့် TIM2 Control Register ကိုကြည့်ရပါမယ်။

18.4.1 TIMx control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Direction ကို ရွေးဖို့ရန်အတွက် DIR ဆိုတာလေး bit 4 မှာရှိပါတယ်။ နည်းနည်းဆွဲကြည့်လိုက်တော့ -

Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter

Bit 4 မှာပြောထားတာက - DIR ကို 0 ပေးရင် Up Counter ဖြစ်ပြီးတော့ 1 ပေးရင် down counter ဖြစ်ပါတယ်။ Up Counter ဘဲလုပ်လိုက်ပါမယ်။ အဲ့ဒီအတွက်ကြောင့်အောက်ပါ ကုဒ်ကိုရေးပါတယ်။

__clearbit(TIM2->CR1, 4U);

မီးလုံးအပိတ်နဲ့အဖွင့်ရဲ့ကြားထဲကအချိန်ဖြစ်တဲ့ time count ကို 500 ms လိုချင်တယ်။ 500 ms ဆိုတော့ 1 s ရဲ့တစ်ဝက်ပေါ့။

total counts = 500msec * f

= (.5 sec) * 16,000

= 8,000 = 0x1F40

တွက်လို့ရလာတဲ့အချိန်တန်ဖိုးကို TIM2 auto-reload register(ARR) ထဲကို အခုလိုမျိုးထည့်လိုက်ပါတယ်။ TIM2->ARR = 8000;

18.4.12 TIMx auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARR[31:16] (depending on timers)															
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
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

ARR ကို 8000 ပေးထားခဲ့တဲ့အတွက်ကြောင့် 0 ကနေ 8000 အထိ count up လုပ်သွားမှာဖြစ်ပါတယ်။ 8000 ကျော်သွားတာနဲ့ Overflow interrupt ကိုထုတ်ပေးရမှာဖြစ်ပါတယ်။ Count Up လုပ်ခဲ့လို့ Overflow Interrupt ကိုထုတ်ပေးရတာဖြစ်ပါတယ်။ အကယ်၍သာ count down လုပ်ခဲ့ရင် Underflow interrupt ကိုထုတ်ပေးရမှာဖြစ်ပါတယ်။ ဒါကိုလည်း timer နဲ့ပတ်သတ်တဲ့ concept တွေရှင်းပြခဲ့တဲ့နေရာမှာပြောခဲ့ပြီးသားဖြစ်ပါတယ်။

```
__setbit(TIM2->DIER, 0U);
```

ဒီ code လေးကိုရေးပြီးတော့ Overflow Interrupt ကိုထုတ်ပေးလိုက်ပါပြီ။ ဒီ ကုဒ်ကိုဘာအတွက်ရေးရတယ် ဆိုတာကိုရှင်းပြပါရစေ။ Interrupt Enable Register ကို enable လုပ်ဖို့ရန်အတွက် ဒီကုဒ်ကိုရေးပါတယ်။

18.4.4 TIMx DMA/Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

အခဲ့ဒီ datasheet ဇယားကနေ bit 0 ကိုသွားကြည့်ပါမယ်။

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

Bit 0 မှာရေးထားတာက 0 ဆိုရင် interrupt ကို disable လုပ်မှာဖြစ်ပြီးတော့ 1 ဆိုရင် interrupt ကို enable လုပ်မှာဖြစ်ပါတယ်။ ကျွန်တော်တို့က enable လုပ်မှာဖြစ်တဲ့အတွက်ကြောင့် 1 ပေးခဲ့ပါတယ်။

Timer 2 interrupt ဟာ NVIC ရဲ့ IRQ-6 မှာလက်ခံရရှိပါတယ်။ ဒါကြောင့် Interrupt ကိုသိရှိနိုင်ဖို့ရန်အတွက် NVIC ရဲ့ IRQ6 ကို enable လုပ်လိုက်ပါတယ်။

```
NVIC_EnableIRQ(TIM2_IRQn);
```

ပြီးရင် Timer-2 ကိုစခိုင်းလိုက်မှာဖြစ်ပါတယ်။ ဒါကြောင့် Timer 2 ရဲ့ Control Register 1(CR1) ကိုသွားကြည့်ပါမယ်။

18.4.1 TIMx control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 0 က counter enable(CEN) ဖြစ်တဲ့အတွက် Bit 0 ကိုသွားဖတ်ကြည့်လိုက်ပါမယ်။

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled

ဖတ်ကြည့်လိုက်တော့ Bit 0 ကို 1 ပေးလိုက်ရင် counter enable လုပ်လိုက်တာဖြစ်ပြီးတော့ 0 ပေးလိုက်ရင် disable လုပ်တယ်ဆိုတာကို မြင်ရမှာဖြစ်ပါတယ်။

```
__setbit(TIM2->CR1, 0U);
```

ပြီးရင် Timer 2 အတွက် Interrupt Handler ကိုရေးပေးရမှာဖြစ်ပါတယ်။ Timer 2 ရဲ့ status register ကိုသွားကြည့်ပါမယ်။

18.4.5 TIMx status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Reserved		TIF	Res	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

အဲဒီမှာ bit 0 က Update Interrupt Flag(UIF) ဖြစ်ပါတယ်။ သူ့ကို clear လုပ်ဖို့ရန်အတွက် အခုလိုရေးလိုက်ပါတယ်။

```
__clearbit(TIM2->SR, 0U);
```

GPIO Output အကြောင်း အရင်ကပြောခဲ့ပြီးသားဖြစ်တဲ့အတွက်ကြောင့် code ကိုအပြည့်အစုံပြပါတော့မယ်။ ရှင်းပြပြီးသားဖြစ်တဲ့အတွက်ကြောင့် နားလည်လိမ့်မယ်လို့ယူဆပါတယ်။

```
#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 initLed(void);

#define	UP_COUNTER	1
int main () {
	initLed();
	__setbit(RCC->APB1ENR, 0U);
	TIM2->PSC = 1000;

	#if (UP_COUNTER)
		__clearbit(TIM2->CR1, 4U);
	#else
		__clearbit(TIM2->CR1, 4U);

	#endif
	TIM2->ARR = 8000;
	__setbit(TIM2->DIER, 0U);
	NVIC_EnableIRQ(TIM2_IRQn);
	 __setbit(TIM2->CR1, 0U);
	for (;;) {}
}

#ifdef	__cplusplus
	extern "C" {

#endif

void TIM2_IRQHandler (void) {
	__clearbit(TIM2->SR, 0U);
```

```
                __togglebit(GPIOD->ODR, 15);
        }
        #ifdef   __cplusplus
}
#endif
static void initLed(void) {
         __setbit(RCC->AHB1ENR, 3);
        __setbit(GPIOD->MODER, 30);
         __clearbit(GPIOD->MODER, 31);
        __clearbit(GPIOD->OTYPER, 15);
        __setbit(GPIOD->OSPEEDR, 30);
        __clearbit(GPIOD->OSPEEDR, 31);
         __clearbit(GPIOD->PUPDR, 30);
         __clearbit(GPIOD->PUPDR, 31);
}
```