Timer Değer Okuma

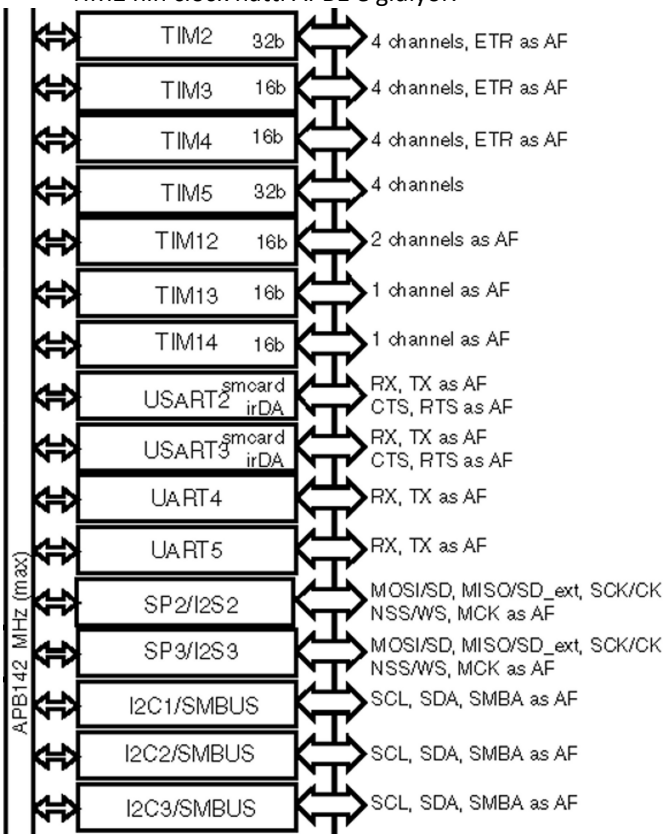
25 Aralık 2021 Cumartesi 00:59

Timer Değer Okuma

- REGİSTER

Konfigürasyon Kısmı

- TIM2'nin clock hattı APB1'e gidiyor.



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
Reserved		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	

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

0.biti 1 yapıyoruz.

Bit 0 **TIM2EN:** TIM2 clock enable
Set and cleared by software.
0: TIM2 clock disabled
1: TIM2 clock enabled

```
RCC->APB1ENR |= 0x00000001;
```

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

- Sayma yapacağımdan saymayı başlatmak için 0.biti aktif ediyoruz. Sayma başlayacağından bu işlem fonksiyonun en son satırında olmalı.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

```
TIM2->CR1 |= 1 << 0;
```

- Saymanın yönünü belirliyoruz. Biz yukarı doğru saymasını istiyoruz.

Bit 4 **DIR**: Direction

0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

```
TIM2->CR1 |= 0 << 4;
```

- 5. ve 6.biti 0 yapıyoruz.

Bits 6:5 **CMS**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

- Sistem clock 168MHz ayarlamıştık ve TIM2 clock veri yolu ise 42MHz'dir fakat bunun 2 katı değeri alıyorlardı yani 84MHz'dir. Biz bunu bölmek istiyorsak ayarlayabiliyoruz. Biz bölmüyoruz.

Bits 9:8 **CKD**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (ETR, TIx),

00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 \times t_{CK\_INT}$
10: $t_{DTS} = 4 \times t_{CK\_INT}$
11: Reserved

- 8. ve 9.biti 0 yapıyoruz.

```
TIM2->CR1 |= 0 << 8;
```

TIMx slave mode control register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

- Slave mode çalışmayacağız.

Bits 2:0 **SMS:** Slave mode selection

Bits 2:0 **SMS:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.
000: Slave mode disabled - if CEN = '1 then the prescaler is clocked directly by the internal clock.
001: Encoder mode 1 - Counter counts up/down on TI2FP1 edge depending on TI1FP2 level.
010: Encoder mode 2 - Counter counts up/down on TI1FP2 edge depending on TI2FP1 level.
011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.
100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.
101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.
111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

The clock of the slave timer must be enabled prior to receiving events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

```
TIM2->SMCR |= 0 << 0;
```

TIMx event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

- Sayma dolduğunda sıfırlaması için 1 yaparız. 0.biti 1 yapıyoruz.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

```
TIM2->EGR |= 1 << 0;
```

TIMx prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

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

- Prescalaer bizim sayısının en üst seviyesini belirler.
- Sayma işlemi 0'dan başlamayıp 1'den başladığından 1 eksiğini alarak yazarız.
- TIM2 clock hızı 84MHz olduğundan bunu kaça bölmeliyim diye soruyoruz. Bu clock hız için 42000'e bölüyoruz. Bu sayı da 41999 sayısı yapıyor.

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in "reset mode").

```
TIM2->PSC |= 41999;
```

- 84MHz'i 42000'e böldüğümüzde 2000 sayısı yapıyor. Bu değer auto-reload oluyor. Bunun 1 eksiğini yazıyoruz.

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

Bits 31:16 **ARR[31:16]**: High auto-reload value (on TIM2 and TIM5).

Bits 15:0 **ARR[15:0]**: Auto-reload value
ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 18.3.1: Time-base unit* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

```
TIM2->ARR |= 1999;
```

- Böylece 1 sn'de 2000'e kadar sayıyor.
- RCC ve TIM için yazdığımız fonksiyonlar aşağıdaki gibidir.

```
5  void RCC_Config(void)
6  {
7      RCC->CR |= 0x00010000;              //HSEON
8      while(!(RCC->CR & 0x00020000));     //HSERDY
9      RCC->CR |= 0x00080000;              //CSSON
10     RCC->CFGR = 0x00000000;
11     RCC->PLLCFGR |= 0x00400000;         //PLLSRC
12     RCC->PLLCFGR |= 0x00000004;         //PLLM 4
13     RCC->PLLCFGR |= 0x00002A00;         //PLLN 168
14     RCC->PLLCFGR |= 0x00000000;         //PLLP 2
15     RCC->CR |= 0x01000000;              //PLLON
16     while(!(RCC->CR & 0x02000000));     //PLLRDY
17     RCC->CFGR |= 0x00000001;            //SW
18     while(!(RCC->CR & 0x00000001));     //SWS
19     RCC->CFGR |= 0x00000000;            //HPRE AHB 1
20     RCC->CFGR |= 0x00001400;            //PPRE1 APB1 4
21     RCC->CFGR |= 0x00008000;            //PPRE2 APB2 2
22     RCC->CIR |= 0x00080000;             //HSERDYC
23     RCC->CIR |= 0x00800000;             //CSSC
24 }
```

```
26 void TIM_Config(void)
27 {
28     RCC->APB1ENR |= 0x00000001;         //TIM2EN
29
30     TIM2->CR1 |= 0 << 4;                //DIR Counter used as up counter
31     TIM2->CR1 |= 0 << 5;                //CMS Edge-aligned mode
32     TIM2->CR1 |= 0 << 8;                //tDTS = tCK_INT 84MHz
33     TIM2->SMCR |= 0 << 0;               //SMS Slave mode disabled
34     TIM2->EGR |= 1 << 0;                //UG Update generation
35     TIM2->PSC |= 41999;                 //PSC Prescaler value
36     TIM2->ARR |= 1999;                  //ARR Auto-reload value
37     TIM2->CR1 |= 1 << 0;                //CEN Counter enabled
38 }
```

Kod Kısmı

TIMx counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT[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
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- CNT registerı sayma işlemini burada tutuyor.
- Değişken atayıp bu değişkene CNT registerına eşitliyoruz.

```
3  uint16_t count = 0;

40 int main(void)
41 {
42     RCC_Config();
43     TIM_Config();
44
45   while (1)
46   {
47       count = TIM2->CNT;
48
49   }
50 }
```

Variable Name	Address/Expression	Read Value
count	0x2000002c	1838