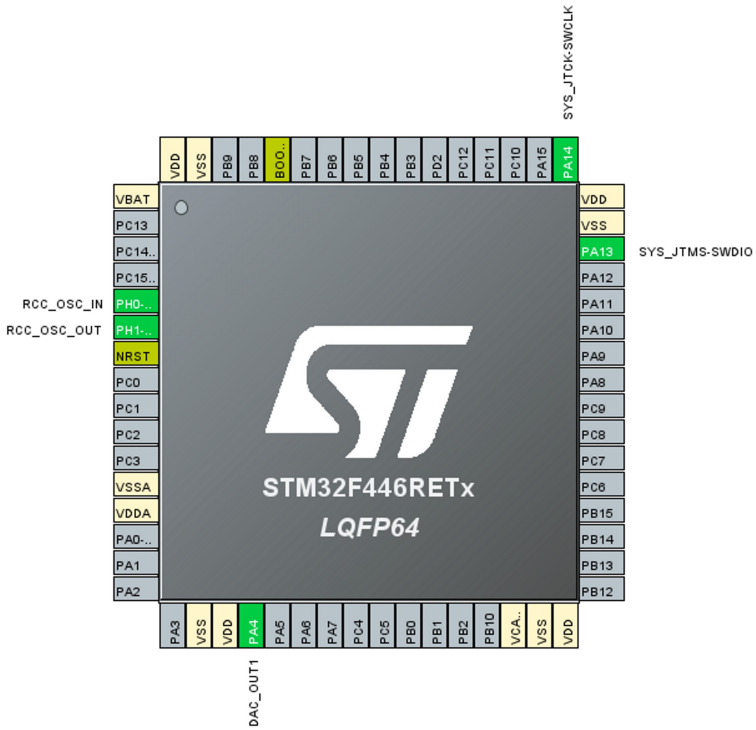
DAC Kullanımı

25 Aralık 2021 Cumartesi 00:57

DAC Kullanımı

- HAL

Konfigürasyon Kısmı



Pin N...	Signal on...	GPIO out...	GPIO mo...	GPIO Pul...	Maximu...	User Label	Modified
PA4	DAC_O...	n/a	Analog ...	No pull-u...	n/a		☐

- DAC kısmından OUT1 seçiyoruz.

☑ OUT1 Configuration

☐ OUT2 Configuration

☐ External Trigger

- Output Buffer, gürültü engelleme; Trigger tetiklemedir.
  Biz bunları aynı şekilde bırakıyoruz.

DAC Out1 Settings

Output Buffer	Enable
Trigger	None

Kod Kısmı

- DAC için toplam iki kanal var.

```
18      [..]
19        *** DAC Channels ***
20        ====================
21      [..]
22      STM32F4 devices integrate two 12-bit Digital Analog Converters
23
24      The 2 converters (i.e. channel1 & channel2)
25      can be used independently or simultaneously (dual mode):
26        (#) DAC channel1 with DAC_OUT1 (PA4) as output
27        (#) DAC channel2 with DAC_OUT2 (PA5) as output
```

- DAC kullanmak için iki modumuz var. Biri Polling diğeri DMA'dır.
- Biz Polling Mode kullanıyoruz.

```
110      *** Polling mode IO operation ***
111      =================================
112      [..]
113        (+) Start the DAC peripheral using HAL_DAC_Start()
114        (+) To read the DAC last data output value, use the HAL_DAC_GetValue() function.
115        (+) Stop the DAC peripheral using HAL_DAC_Stop()
```

```
435 HAL_StatusTypeDef HAL_DAC_Start(DAC_HandleTypeDef *hdac, uint32_t Channel)
```

- DAC için değer verme yapacağımızdan GetValue değil SetValue kullanacağız. Çıkışın voltaj değeri ile oynamış oluyoruz.

```
759 /**
760   * @brief  Set the specified data holding register value for DAC channel.
761   * @param  hdac pointer to a DAC_HandleTypeDef structure that contains
762   *         the configuration information for the specified DAC.
763   * @param  Channel The selected DAC channel.
764   *          This parameter can be one of the following values:
765   *            @arg DAC_CHANNEL_1: DAC Channel1 selected
766   *            @arg DAC_CHANNEL_2: DAC Channel2 selected
767   * @param  Alignment Specifies the data alignment.
768   *          This parameter can be one of the following values:
769   *            @arg DAC_ALIGN_8B_R: 8bit right data alignment selected
770   *            @arg DAC_ALIGN_12B_L: 12bit left data alignment selected
771   *            @arg DAC_ALIGN_12B_R: 12bit right data alignment selected
772   * @param  Data Data to be loaded in the selected data holding register.
773   * @retval HAL status
774   */
775 HAL_StatusTypeDef HAL_DAC_SetValue(DAC_HandleTypeDef *hdac, uint32_t Channel, uint32_t Alignment, uint32_t Data)
```

- Data kısmına 12 bit çalıştığımızdan 0-4095 arasında değer yazabiliriz.

```
23 /* Private includes ----------
24 /* USER CODE BEGIN Includes */
25 int i=0;
26 /* USER CODE END Includes */
```

```
91   /* USER CODE BEGIN 2 */
92 HAL_DAC_Start(&hdac, DAC_CHANNEL_1);
93   /* USER CODE END 2 */
```

```
95   /* Infinite loop */
96   /* USER CODE BEGIN WHILE */
97   while (1)
98   {
99     /* USER CODE END WHILE */
100
101    /* USER CODE BEGIN 3 */
102      do
103      {
104          HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, i);
105          HAL_Delay(1);
106          i++;
107      }
108      while(i<4096);
109          i=0;
110  }
111  /* USER CODE END 3 */
```

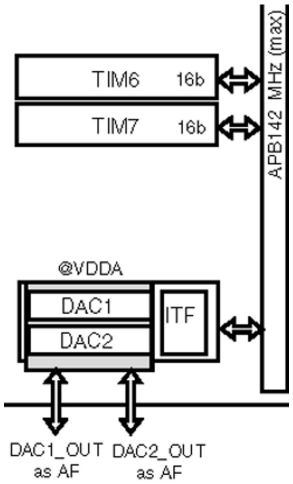
➢ REGİSTER

Konfigürasyon Kısmı

- RCC için 5.satırda 0x00030000 yerine 0x00010000 yazdık.
  HSERDY kısmı sadece okunabilir olduğundan o biti 1 yapmıyoruz.

```
void RCC_Config(void)
{
    RCC->CR |= 0x00010000;              //HSEON
    while(!(RCC->CR & 0x00020000));     //HSERDY
    RCC->CR |= 0x00080000;              //CSSON
    RCC->CFGR = 0x00000000;
    RCC->PLLCFGR |= 0x00400000;         //PLLSRC
    RCC->PLLCFGR |= 0x00000004;         //PLLM 4
    RCC->PLLCFGR |= 0x00002A00;         //PLLN 168
    RCC->PLLCFGR |= 0x00000000;         //PLLP 2
    RCC->CR |= 0x01000000;              //PLLON
    while(!(RCC->CR & 0x02000000));     //PLLRDY
    RCC->CFGR |= 0x00000001;            //SW
    while(!(RCC->CR & 0x00000001));     //SWS
}
```

- ADC'nin clock hattı APB1'e gidiyor.



7.3.13 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
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

- 29.pini 1 yapıyoruz.

Bit 29 **DACEN:** DAC interface clock enable
Set and cleared by software.
0: DAC interface clock disabled
1: DAC interface clock enable

```
RCC->APB1ENR |= 0x20000000;          //DACEN
```

DAC control register (DAC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[2:0]			TEN2	BOFF2	EN2
		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
Reserved		DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- DAC işlemi için channel1 kullanacağımızdan 0.biti aktif etmemiz gerekiyor.

Bit 0 **EN1**: DAC channel1 enable

This bit is set and cleared by software to enable/disable DAC channel1.
0: DAC channel1 disabled
1: DAC channel1 enabled

```
DAC->CR |= 0x00000001;
```

- Yazılımsal tetikleme kullanmadığımızdan 0. biti 0 yapıyoruz.

DAC software trigger register (DAC_SWTRIGR)

Address offset: 0x04
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														SWTRIG2	SWTRIG1
														w	w

Bit 0 **SWTRIG1**: DAC channel1 software trigger

This bit is set and cleared by software to enable/disable the software trigger.
0: Software trigger disabled
1: Software trigger enabled

Note: This bit is cleared by hardware (one APB1 clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.

```
DAC->SWTRIGR |= 0x00000000;
```

- Channel1 için 12 bit sağa ya da sola veya 8 bit sağa kaydedebiliriz.

DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

Address offset: 0x08
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DACC1DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

- 12 bit sağa kaydedeceğiz.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

RCC AHB1 peripheral clock enable register (RCC_AHB1ENR)

Address offset: 0x30

Reset value: 0x0010 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
Reser-ved	OTGHS ULPIEN	OTGHSEN	ETHMACPTPEN	ETHMACRXEN	ETHMACTXEN	ETHMACEN	Reserved		DMA2EN	DMA1EN	CCMDATARAMEN	Res.	BKPSRAMEN	Reserved	
	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
Reserved			CRCEN	Reserved			GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
			rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

- DAC1 A4 pini olduğundan portunu sadece A portunu aktif ediyoruz.

```
RCC->AHB1ENR |= 0x00000001;
```

- RCC ve DAC1 için yazdığımız fonksiyonlar aşağıdaki gibidir.

```
10 void RCC_Config(void)
11 {
12     RCC->CR |= 0x00010000;              //HSEON
13     while(!(RCC->CR & 0x00020000));     //HSERDY
14     RCC->CR |= 0x00080000;              //CSSON
15     RCC->CFGR = 0x00000000;
16     RCC->PLLCFGR |= 0x00400000;         //PLLSRC
17     RCC->PLLCFGR |= 0x00000004;         //PLLM 4
18     RCC->PLLCFGR |= 0x00002A00;         //PLLN 168
19     RCC->PLLCFGR |= 0x00000000;         //PLLP 2
20     RCC->CR |= 0x01000000;              //PLLON
21     while(!(RCC->CR & 0x02000000));     //PLLRDY
22     RCC->CFGR |= 0x00000001;            //SW
23     while(!(RCC->CR & 0x00000001));     //SWS
24 }

26 void DAC1_Config(void)
27 {
28     RCC->APB1ENR |= 0x20000000;         //DACEN
29     RCC->AHB1ENR |= 0x00000001;         //A clock enable
30
31     DAC->CR |= 0x00000001;              //DAC channel1 enabled
32     DAC->SWTRIGR |= 0x00000000;         //Software trigger disabled
33     DAC->DHR12R1 |= 0x00000000;         //DAC channel1 12-bit right-aligned data
34 }
```

Kod Kısmı

```c
int i=0;

void delay(uint32_t time)
{
    while(time--);
}
```

- Her i değeri arttığında i değeri kadar kaydetmiş oluyor.
- A4 pinine led bağladığımızda i değeri arttıkça voltaj değeri artacağından parlaklık artar.
- 4095 olduğunda yani 0x00000FFF olduğunda tam parlaklıkta yanar.

```c
int main(void)
{
    RCC_Config();
    SystemCoreClockUpdate();
    DAC1_Config();

  while (1)
  {
      for(;i<4096; i++)
      {
          DAC->DHR12R1 |= i;
          delay(16800);
      }
      i=0;
  }
}
```