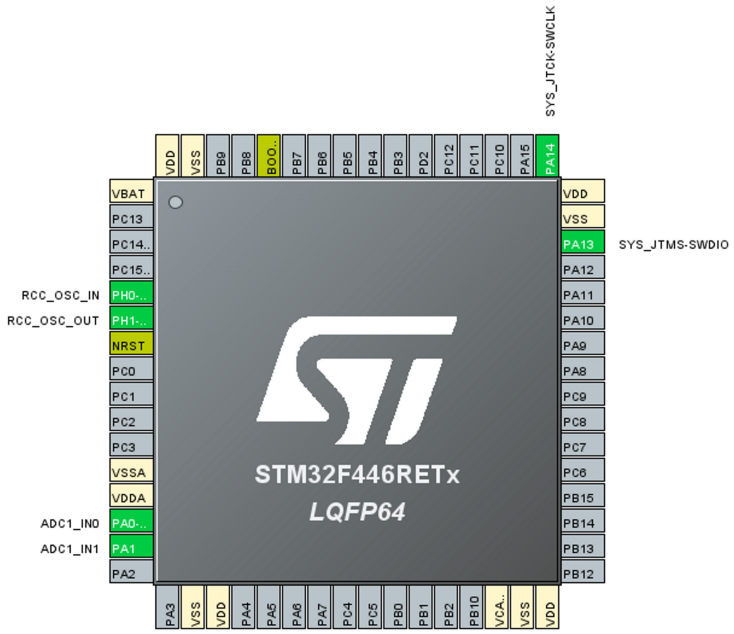
DMA ile ADC Değer Okuma

25 Aralık 2021 Cumartesi 00:58

DMA ile ADC Değer Okuma

- HAL

Konfigürasyon Kısmı



Pin Na...	Signal on P...	GPIO outp...	GPIO mode	GPIO Pull-u...	Maximum o...	User Label	Modified
PA0-WKUP	ADC1_IN0	n/a	Analog mo...	No pull-up ...	n/a		☐
PA1	ADC1_IN1	n/a	Analog mo...	No pull-up ...	n/a		☐

ADC_Settings

Clock Prescaler	PCLK2 divided by 4
Resolution	12 bits (15 ADC Clock cycles)
Data Alignment	Right alignment
Scan Conversion Mode	Enabled
Continuous Conversion Mode	Enabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Enabled
End Of Conversion Selection	EOC flag at the end of single channel conversion

- DMA ayarları için Add diyoruz. Ardından Stream, Direction ve Priority için seçenekleri seçiyoruz. Daha sonra Direction kısmından seçtiğimiz Peripheral ve Memory için adres değişikliği için Increment Adress için işaretliyoruz. Ardından gönderilecek data boyutu belirliyoruz. 12bit çalıştığımızdan 16 bitlik olan Half Word seçimi yapıyoruz.

DMA Request	Stream	Direction	Priority
ADC1	DMA2 Stream 0	Peripheral To Memory	Very High

Add Delete

DMA Request Settings

		Peripheral	Memory
Mode	Circular	Increment Address	☐ ☑
Use Fifo ☐	Threshold	Data Width	Half Word / Half Word
		Burst Size	

Kod Kısmı

```
107       *** DMA mode IO operation ***
108       ==============================
109       [..]
110         (+) Start the ADC peripheral using HAL_ADC_Start_DMA(), at this stage the user specify the length
```

```
107     *** DMA mode IO operation ***
108     ==============================
109     [..]
110       (+) Start the ADC peripheral using HAL_ADC_Start_DMA(), at this stage the user specify the length
111           of data to be transferred at each end of conversion
112       (+) At The end of data transfer by HAL_ADC_ConvCpltCallback() function is executed and user can
113           add his own code by customization of function pointer HAL_ADC_ConvCpltCallback
114       (+) In case of transfer Error, HAL_ADC_ErrorCallback() function is executed and user can
115           add his own code by customization of function pointer HAL_ADC_ErrorCallback
116       (+) Stop the ADC peripheral using HAL_ADC_Stop_DMA()
```

- pData kısmına veriyi tutacağımız değişkeni yazıyoruz. Bizim değişken 16 bit olarak tanımladığımızdan dönüştürmemiz gerekiyor.
- Length ile kaç tane veri okuyacağımızı yazıyoruz.

```
1354 /**
1355   * @brief  Enables ADC DMA request after last transfer (Single-ADC mode) and enables ADC
1356   *         peripheral
1357   * @param  hadc pointer to a ADC_HandleTypeDef structure that contains
1358   *         the configuration information for the specified ADC.
1359   * @param  pData The destination Buffer address.
1360   * @param  Length The length of data to be transferred from ADC peripheral to memory.
1361   * @retval HAL status
1362   */
1363 HAL_StatusTypeDef HAL_ADC_Start_DMA(ADC_HandleTypeDef* hadc, uint32_t* pData, uint32_t Length)
```

```
46 /* USER CODE BEGIN PV */
47 uint16_t adc_value[2];
48 /* USER CODE END PV */
```

```
94   /* USER CODE BEGIN 2 */
95   HAL_ADC_Start_DMA(&hadc1, (uint32_t*)adc_value, 2);
96   /* USER CODE END 2 */
```

➢ REGİSTER

Konfigürasyon Kısmı

- RCC için eklemeler yaptık.

RCC clock configuration register (RCC_CFGR)

Address offset: 0x08

Reset value: 0x0000 0000

Access: 0 ≤ wait state ≤ 2, word, half-word and byte access

1 or 2 wait states inserted only if the access occurs during a clock source switch.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCO2		MCO2 PRE[2:0]			MCO1 PRE[2:0]			I2SSCR	MCO1		RTCPRE[4:0]				
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
PPRE2[2:0]			PPRE1[2:0]			Reserved		HPRE[3:0]				SWS1	SWS0	SW1	SW0
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	r	r	rw	rw

- AHB, APB1 ve APB2 için clock hızını böldük.

Bits 7:4 **HPRE:** AHB prescaler

Set and cleared by software to control AHB clock division factor.

Caution: The clocks are divided with the new prescaler factor from 1 to 16 AHB cycles after HPRE write.

Caution: The AHB clock frequency must be at least 25 MHz when the Ethernet is used.

0xxx: system clock not divided
1000: system clock divided by 2
1001: system clock divided by 4
1010: system clock divided by 8
1011: system clock divided by 16
1100: system clock divided by 64
1101: system clock divided by 128
1110: system clock divided by 256
1111: system clock divided by 512

```
RCC->CFGR |= 0x00000000;
```

Bits 12:10 **PPRE1:** APB Low speed prescaler (APB1)

Set and cleared by software to control APB low-speed clock division factor.

Caution: The software has to set these bits correctly not to exceed 42 MHz on this domain. The clocks are divided with the new prescaler factor from 1 to 16 AHB cycles after PPRE1 write.

0xx: AHB clock not divided
100: AHB clock divided by 2
101: AHB clock divided by 4
110: AHB clock divided by 8
111: AHB clock divided by 16

```
RCC->CFGR |= 0x00001400;
```

Bits 15:13 **PPRE2:** APB high-speed prescaler (APB2)

Set and cleared by software to control APB high-speed clock division factor.

Caution: The software has to set these bits correctly not to exceed 84 MHz on this domain. The clocks are divided with the new prescaler factor from 1 to 16 AHB cycles after PPRE2 write.

0xx: AHB clock not divided
100: AHB clock divided by 2
101: AHB clock divided by 4
110: AHB clock divided by 8
111: AHB clock divided by 16

```
RCC->CFGR |= 0x00008000;
```

RCC clock interrupt register (RCC_CIR)

Address offset: 0x0C

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								CSSC	Reserved	PLLI2SRDYC	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC
								w		w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		PLLI2SRDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	Reserved	PLLI2SRDYF	PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
		rw	rw	rw	rw	rw	rw	r		r	r	r	r	r	r

- Kaldırılan bayrakları temizliyoruz.

Bit 19 **HSERDYC:** HSE ready interrupt clear

This bit is set by software to clear the HSERDYF flag.
0: No effect
1: HSERDYF cleared

```
RCC->CIR |= 0x00080000;
```

Bit 23 **CSSC:** Clock security system interrupt clear

This bit is set by software to clear the CSSF flag.
0: No effect
1: Clear CSSF flag

```
RCC->CIR |= 0x00800000;
```

- RCC için yazdığımız fonksiyon aşağıdaki gibidir.

```c
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
    RCC->CFGR |= 0x00000000;            //HPRE AHB 1
    RCC->CFGR |= 0x00001400;            //PPRE1 APB1 4
    RCC->CFGR |= 0x00008000;            //PPRE2 APB2 2
    RCC->CIR |= 0x00080000;             //HSERDYC
    RCC->CIR |= 0x00800000;             //CSSC
}
```

- GPIO için yazdığımız fonksiyon aşağıdaki gibidir.

```c
void GPIO_Config(void)
{
    RCC->AHB1ENR |= 0x00000001;         //A clock enable

    GPIOA->MODER |= 0x00000003;         //PA0 Analog mode
    GPIOA->OSPEEDR |= 0x00000003;       //Very high speed
}
```

ADC control register 1 (ADC_CR1)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					OVRIE	RES		AWDEN	JAWDEN	Reserved					
					rw	rw	rw	rw	rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM[2:0]			JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- Çoklu ADC okuması yapılacaksa scan yani tarama aktif edilmesi gerekir.

Bit 8 **SCAN:** Scan mode

This bit is set and cleared by software to enable/disable the Scan mode. In Scan mode, the inputs selected through the ADC_SQRx or ADC_JSQRx registers are converted.
0: Scan mode disabled
1: Scan mode enabled

Note: An EOC interrupt is generated if the EOCIE bit is set:
- *At the end of each regular group sequence if the EOCS bit is cleared to 0*
- *At the end of each regular channel conversion if the EOCS bit is set to 1*

Note: A JEOC interrupt is generated only on the end of conversion of the last channel if the JEOCIE bit is set.

```c
ADC1->CR1 |= 1 << 8;
```

- Çözünürlük için 12-bit kullanıyoruz.

Bits 25:24 **RES[1:0]:** Resolution

These bits are written by software to select the resolution of the conversion.
00: 12-bit (15 ADCCLK cycles)
01: 10-bit (13 ADCCLK cycles)
10: 8-bit (11 ADCCLK cycles)
11: 6-bit (9 ADCCLK cycles)

```c
ADC1->CR1 |= 0 << 24;
```

ADC control register 2 (ADC_CR2)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved	SWSTART	EXTEN		EXTSEL[3:0]				reserved	JSWSTART	JEXTEN		JEXTSEL[3:0]			
	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				ALIGN	EOCS	DDS	DMA	Reserved						CONT	ADON
				rw	rw	rw	rw							rw	rw

- ADC aktif etmek için ilk bit 1 yapıldı.

Bit 0 **ADON:** A/D Converter ON / OFF

This bit is set and cleared by software.

Note: 0: Disable ADC conversion and go to power down mode
1: Enable ADC

- Sürekli okuma yapmasını istiyoruz.

Bit 1 **CONT:** Continuous conversion

This bit is set and cleared by software. If it is set, conversion takes place continuously until it is cleared.
0: Single conversion mode
1: Continuous conversion mode

- DMA mod ile çalışacağız.

Bit 8 **DMA:** Direct memory access mode (for single ADC mode)

This bit is set and cleared by software. Refer to the DMA controller chapter for more details.
0: DMA mode disabled
1: DMA mode enabled

- DMA'nın sürekli çalışması için 1 yapıyorum.

Bit 9 **DDS:** DMA disable selection (for single ADC mode)

This bit is set and cleared by software.
0: No new DMA request is issued after the last transfer (as configured in the DMA controller)
1: DMA requests are issued as long as data are converted and DMA=1

- 10.bit 1 yapılır.

Bit 10 **EOCS:** End of conversion selection

This bit is set and cleared by software.
0: The EOC bit is set at the end of each sequence of regular conversions. Overrun detection is enabled only if DMA=1.
1: The EOC bit is set at the end of each regular conversion. Overrun detection is enabled.

- Çevrim başlaması için 30.biti 1 yapıyoruz.

Bit 30 **SWSTART:** Start conversion of regular channels

This bit is set by software to start conversion and cleared by hardware as soon as the conversion starts.
0: Reset state
1: Starts conversion of regular channels

Note: This bit can be set only when ADON = 1 otherwise no conversion is launched.

```
ADC1->CR2 |= 1 << 0;
ADC1->CR2 |= 1 << 1;
ADC1->CR2 |= 1 << 8;
ADC1->CR2 |= 1 << 9;
ADC1->CR2 |= 1 << 10;
ADC1->CR2 |= 1 << 30;
```

- Kaç farklı kanaldan çevrim yapacağımızı belirtmemiz gerekiyor. Biz sadece A portun 0.bitinden yani 1 kanal'dan okuma yapacağım.

ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								L[3:0]				SQ16[4:1]			
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								L[3:0]				SQ16[4:1]			
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ16_0	SQ15[4:0]					SQ14[4:0]					SQ13[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

ADC regular sequence register 2 (ADC_SQR2)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ12[4:0]					SQ11[4:0]					SQ10[4:1]			
		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
SQ10_0	SQ9[4:0]					SQ8[4:0]					SQ7[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ6[4:0]					SQ5[4:0]					SQ4[4:1]			
		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
SQ4_0	SQ3[4:0]					SQ2[4:0]					SQ1[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- SQR1 için 20.biti 0 yaparak 1 tane kanal olduğunu belirtiyoruz.

Bits 23:20 **L[3:0]:** Regular channel sequence length

These bits are written by software to define the total number of conversions in the regular channel conversion sequence.
0000: 1 conversion
0001: 2 conversions
...
1111: 16 conversions

```
ADC1->SQR1 |= 0 << 20;
```

- SQR3 için SQ1 kısmına kanal numarasını yazıyoruz.

Bits 4:0 **SQ1[4:0]:** 1st conversion in regular sequence

```
ADC1->SQR3 |= 0 << 0;
```

ADC common control register (ADC_CCR)

Address offset: 0x04 (this offset address is relative to ADC1 base address + 0x300)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								TSVREFE	VBATE	Reserved				ADCPRE	
								rw	rw					rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMA[1:0]		DDS	Res.	DELAY[3:0]				Reserved			MULTI[4:0]				
rw	rw	rw		rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 17:16 **ADCPRE:** ADC prescaler

Set and cleared by software to select the frequency of the clock to the ADC. The clock is common for all the ADCs.

Note: 00: PCLK2 divided by 2
01: PCLK2 divided by 4
10: PCLK2 divided by 6
11: PCLK2 divided by 8

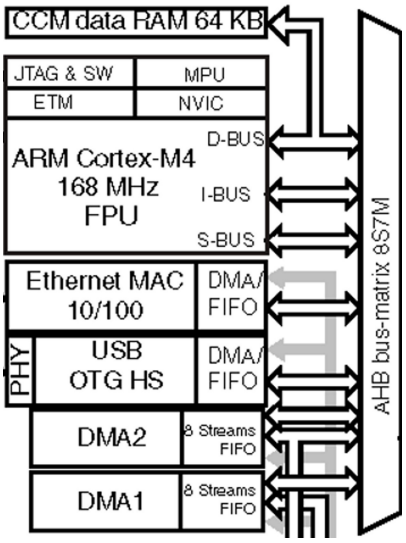
```
ADC->CCR |= 0x00010000;
```

- ADC için yazdığımız fonksiyon aşağıdaki gibidir.

```
void ADC_Config(void)
{
    RCC->APB2ENR |= 0x00000100;          //ADC1

    ADC1->CR1 |= 1 << 8;                 //SCAN
    ADC1->CR1 |= 0 << 24;                //RES 12-bit
    ADC1->CR2 |= 1 << 0;                 //ADON
    ADC1->CR2 |= 1 << 1;                 //CONT
    ADC1->CR2 |= 1 << 8;                 //DMA
    ADC1->CR2 |= 1 << 9;                 //DDS
    ADC1->CR2 |= 1 << 10;                //EOCS
    ADC1->CR2 |= 1 << 30;                //SWSTART
    ADC1->SMPR2 |= 0x00000003;           //SMP0 56 cycles
    ADC1->SQR1 |= 0 << 20;               //1 conversion
    ADC1->SQR3 |= 0 << 0;                //SQ1
    ADC->CCR |= 0x00010000;              //ADCPRE 4
}
```

- DMA'nın clock hattı AHB'ye gidiyor.



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

Bit 22 **DMA2EN:** DMA2 clock enable
Set and cleared by software.
0: DMA2 clock disabled
1: DMA2 clock enabled

- 22.pini 1 yapıyoruz.

```
RCC->AHB1ENR |= 0x00400000;
```

- ADC için DMA2 kullanıyoruz. Tabloya baktığımızda biz burada Stream 4'ü seçiyoruz.

DMA1 request mapping

Peripheral requests	Stream 0	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7
Channel 0	SPI3_RX	-	SPI3_RX	SPI2_RX	SPI2_TX	SPI3_TX	-	SPI3_TX
Channel 1	I2C1_RX	-	TIM7_UP	-	TIM7_UP	I2C1_RX	I2C1_TX	I2C1_TX
Channel 2	TIM4_CH1	-	I2S3_EXT_RX	TIM4_CH2	I2S2_EXT_TX	I2S3_EXT_TX	TIM4_UP	TIM4_CH3
Channel 3	I2S3_EXT_RX	TIM2_UP TIM2_CH3	I2C3_RX	I2S2_EXT_RX	I2C3_TX	TIM2_CH1	TIM2_CH2 TIM2_CH4	TIM2_UP TIM2_CH4
Channel 4	UART5_RX	USART3_RX	UART4_RX	USART3_TX	UART4_TX	USART2_RX	USART2_TX	UART5_TX
Channel 5	UART8_TX(1)	UART7_TX(1)	TIM3_CH4 TIM3_UP	UART7_RX(1)	TIM3_CH1 TIM3_TRIG	TIM3_CH2	UART8_RX(1)	TIM3_CH3
Channel 6	TIM5_CH3 TIM5_UP	TIM5_CH4 TIM5_TRIG	TIM5_CH1	TIM5_CH4 TIM5_TRIG	TIM5_CH2	-	TIM5_UP	-
Channel 7	-	TIM6_UP	I2C2_RX	I2C2_RX	USART3_TX	DAC1	DAC2	I2C2_TX

1. These requests are available on STM32F42xxx and STM32F43xxx only.

DMA2 request mapping

Peripheral requests	Stream 0	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7
Channel 0	ADC1	SAI1_A(1)	TIM8_CH1 TIM8_CH2 TIM8_CH3	SAI1_A(1)	ADC1	SAI1_B(1)	TIM1_CH1 TIM1_CH2 TIM1_CH3	-
Channel 1	-	DCMI	ADC2	ADC2	SAI1_B(1)	SPI6_TX(1)	SPI6_RX(1)	DCMI
Channel 2	ADC3	ADC3	-	SPI5_RX(1)	SPI5_TX(1)	CRYP_OUT	CRYP_IN	HASH_IN
Channel 3	SPI1_RX	-	SPI1_RX	SPI1_TX	-	SPI1_TX	-	-
Channel 4	SPI4_RX(1)	SPI4_TX(1)	USART1_RX	SDIO	-	USART1_RX	SDIO	USART1_TX
Channel 5	-	USART6_RX	USART6_RX	SPI4_RX(1)	SPI4_TX(1)	-	USART6_TX	USART6_TX
Channel 6	TIM1_TRIG	TIM1_CH1	TIM1_CH2	TIM1_CH1	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	-
Channel 7	-	TIM8_UP	TIM8_CH1	TIM8_CH2	TIM8_CH3	SPI5_RX(1)	SPI5_TX(1)	TIM8_CH4 TIM8_TRIG TIM8_COM

1. These requests are available on STM32F42xxx and STM32F43xxx.

DMA stream x configuration register (DMA_SxCR) (x = 0..7)

This register is used to configure the concerned stream.

Address offset: 0x10 + 0x18 × *stream number*

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CHSEL[2:0]			MBURST [1:0]		PBURST[1:0]		Reser-ved	CT	DBM	PL[1:0]	
				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
PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- 0.bit 0 olduğunda konfigürasyon yapmaya imkan tanıyor. Bunun için okuma yapacağız.

Bit 0 **EN**: Stream enable / flag stream ready when read low

This bit is set and cleared by software.

0: Stream disabled

1: Stream enabled

This bit may be cleared by hardware:

- on a DMA end of transfer (stream ready to be configured)
- if a transfer error occurs on the AHB master buses
- when the FIFO threshold on memory AHB port is not compatible with the size of the burst

When this bit is read as 0, the software is allowed to program the Configuration and FIFO bits registers. It is forbidden to write these registers when the EN bit is read as 1.

Note: Before setting EN bit to '1' to start a new transfer, the event flags corresponding to the stream in DMA_LISR or DMA_HISR register must be cleared.

- 1 olduğu sürece 0 olana kadar bekliyoruz. 0 olunca döngüden çıkıp alt satıra geçecek.

```
while((DMA2_Stream4->CR & 0x00000001) == 1);
```

- ADC'den RAM'e yazacağız bunun için Peripheral-to-memory seçiyoruz.

Bits 7:6 **DIR[1:0]**: Data transfer direction

These bits are set and cleared by software.

00: Peripheral-to-memory

01: Memory-to-peripheral

10: Memory-to-memory

11: reserved

These bits are protected and can be written only if EN is '0'.

```
DMA2_Stream4->CR |= 0 << 6;
```

- Sürekli transfer için 8.biti 1 yapıyoruz.

Bit 8 **CIRC**: Circular mode

This bit is set and cleared by software and can be cleared by hardware.

0: Circular mode disabled

1: Circular mode enabled

When the peripheral is the flow controller (bit PFCTRL=1) and the stream is enabled (bit EN=1), then this bit is automatically forced by hardware to 0.

It is automatically forced by hardware to 1 if the DBM bit is set, as soon as the stream is enabled (bit EN ='1').

```
DMA2_Stream4->CR |= 1 << 8;
```

- Adresin değişmesini istemiyoruz. 9.biti 0 yapıyoruz.

Bit 9 **PINC**: Peripheral increment mode

This bit is set and cleared by software.

0: Peripheral address pointer is fixed

1: Peripheral address pointer is incremented after each data transfer (increment is done according to PSIZE)

This bit is protected and can be written only if EN is '0'.

```
DMA2_Stream4->CR |= 0 << 9;
```

Bit 10 **MINC**: Memory increment mode

This bit is set and cleared by software.

0: Memory address pointer is fixed

1: Memory address pointer is incremented after each data transfer (increment is done according to MSIZE)

This bit is protected and can be written only if EN is '0'.

```
DMA2_Stream4->CR |= 1 << 10;
```

- Peripheral ve Memory için data boyutunu 32 bit belirliyoruz.

Bits 12:11 **PSIZE[1:0]**: Peripheral data size

These bits are set and cleared by software.

00: Byte (8-bit)

01: Half-word (16-bit)

10: Word (32-bit)

11: reserved

These bits are protected and can be written only if EN is '0'

Bits 14:13 **MSIZE[1:0]**: Memory data size

These bits are set and cleared by software.
00: byte (8-bit)
01: half-word (16-bit)
10: word (32-bit)
11: reserved
These bits are protected and can be written only if EN is '0'.
In direct mode, MSIZE is forced by hardware to the same value as PSIZE as soon as bit EN = '1'.

```
DMA2_Stream4->CR |= 2 << 11;                    //PSIZE Word (32-bit)
DMA2_Stream4->CR |= 2 << 13;                    //MSIZE Word (32-bit)
```

- Önceliği çok yüksek yapıyoruz.

Bits 17:16 **PL[1:0]**: Priority level

These bits are set and cleared by software.
00: Low
01: Medium
10: High
11: Very high
These bits are protected and can be written only if EN is '0'.

```
DMA2_Stream4->CR |= 3 << 16;
```

- Kanal seçimi için 0.kanal yapıyoruz.

Bits 27:25 **CHSEL[2:0]**: Channel selection

These bits are set and cleared by software.
000: channel 0 selected
001: channel 1 selected
010: channel 2 selected
011: channel 3 selected
100: channel 4 selected
101: channel 5 selected
110: channel 6 selected
111: channel 7 selected
These bits are protected and can be written only if EN is '0'

- Kaç tane çevresel birimden yani ADC'den okuma yaptığımızı belirtiyoruz.

DMA stream x number of data register (DMA_SxNDTR) (x = 0..7)

Address offset: 0x14 + 0x18 × *stream number*

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
NDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **NDT[15:0]**: Number of data items to transfer

Number of data items to be transferred (0 up to 65535). This register can be written only when the stream is disabled. When the stream is enabled, this register is read-only, indicating the remaining data items to be transmitted. This register decrements after each DMA transfer.
Once the transfer has completed, this register can either stay at zero (when the stream is in normal mode) or be reloaded automatically with the previously programmed value in the following cases:

- when the stream is configured in Circular mode.
- when the stream is enabled again by setting EN bit to '1'

If the value of this register is zero, no transaction can be served even if the stream is enabled.

- Biz bir tane birimden yani ADC1'den okuma yapıyoruz.

```
DMA2_Stream4->NDTR |= 1;
```

- Okuma yaptığımız Çevresel birimin adresini yazıyoruz.

DMA stream x peripheral address register (DMA_SxPAR) (x = 0..7)

Address offset: 0x18 + 0x18 × *stream number*

Reset value: 0x0000 0000

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

Bits 31:0 **PAR[31:0]**: Peripheral address

Base address of the peripheral data register from/to which the data will be read/written.

These bits are write-protected and can be written only when bit EN = '0' in the DMA_SxCR register.

DMA transfer.

Once the transfer has completed, this register can either stay at zero (when the stream is in normal mode) or be reloaded automatically with the previously programmed value in the following cases:

- when the stream is configured in Circular mode.
- when the stream is enabled again by setting EN bit to '1'

If the value of this register is zero, no transaction can be served even if the stream is enabled.

ADC regular data register (ADC_DR)

Address offset: 0x4C

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
DATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DATA[15:0]:** Regular data

These bits are read-only. They contain the conversion result from the regular channels. The data are left- or right-aligned as shown in *Figure 48* and *Figure 49*.

```
DMA2_Stream4->PAR |= (uint8_t) &ADC1->DR;
```

- Değişkenin adresini yazıyoruz.

DMA stream x memory 0 address register (DMA_SxM0AR) (x = 0..7)

Address offset: 0x1C + 0x18 × *stream number*

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
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
M0A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **M0A[31:0]**: Memory 0 address

Base address of Memory area 0 from/to which the data will be read/written.

These bits are write-protected. They can be written only if:

- the stream is disabled (bit EN= '0' in the DMA_SxCR register) or
- the stream is enabled (EN='1' in DMA_SxCR register) and bit CT = '1' in the DMA_SxCR register (in Double buffer mode).

```
3 uint8_t adc;
4 uint8_t adc1[8];
```

```
DMA2_Stream4->M0AR |= (uint8_t) &adc1;
```

DMA stream x FIFO control register (DMA_SxFCR) (x = 0..7)

Address offset: 0x24 + 0x24 × *stream number*

Reset value: 0x0000 0021

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								FEIE	Reserved	FS[2:0]			DMDIS	FTH[1:0]	
								rw		r	r	r	rw	rw	rw

Bits 1:0 **FTH[1:0]**: FIFO threshold selection
These bits are set and cleared by software.
00: 1/4 full FIFO
01: 1/2 full FIFO
10: 3/4 full FIFO
11: full FIFO
These bits are not used in the direct mode when the DMIS value is zero.
These bits are protected and can be written only if EN is '0'.

```
DMA2_Stream4->FCR |= 0 << 1;
```

- Stream4'ü fonksiyonun en son satırında aktif ediyoruz.

```
DMA2_Stream4->CR |= 1 << 0;
```

- Adres tanımları fonksiyonun ortalarında yazmak yerine while döngüsünün sonrasında yazdık.
- DMA için yazdığımız fonksiyon aşağıdaki gibidir.

```
53 void DMA_Config(void)
54 {
55     RCC->AHB1ENR |= 0x00400000;          //DMA2
56
57     while((DMA2_Stream4->CR & 0x00000001) == 1);
58     DMA2_Stream4->PAR |= (uint8_t) &ADC1->DR;          //PAR Peripheral address
59     DMA2_Stream4->M0AR |= (uint8_t) &adc1;            //M0A Memory 0 address
60     DMA2_Stream4->CR |= 0 << 6;                       //DIR
61     DMA2_Stream4->CR |= 1 << 8;                       //CIRC
62     DMA2_Stream4->CR |= 0 << 9;                       //PINC
63     DMA2_Stream4->CR |= 1 << 10;
64     DMA2_Stream4->CR |= 2 << 11;                      //PSIZE Word (32-bit)
65     DMA2_Stream4->CR |= 2 << 13;                      //MSIZE Word (32-bit)
66     DMA2_Stream4->CR |= 3 << 16;                      //PL
67     DMA2_Stream4->CR |= 0 << 25;                      //CHSEL
68     DMA2_Stream4->NDTR |= 1;                          //NDT
69     DMA2_Stream4->FCR |= 0 << 1;                      //FTH 1/2 full FIFO
70     DMA2_Stream4->CR |= 1 << 0;                       //EN Stream enabled
71 }
```

Kod Kısmı

- ADC için çevrimi DMA'dan sonra başlatmak için 80.satıra tekrar yazdık.

```
73 int main(void)
74 {
75     RCC_Config();
76     GPIO_Config();
77     ADC_Config();
78     DMA_Config();
79     //ADC1->CR2 |= ADC_CR2_SWSTART;
80     ADC1->CR2 |= 0x40000000;
81
82   while (1)
83   {
84
85   }
86 }
```