

TI DSP, MCU, Xilinx Zynq FPGA

프로그래밍 전문가 과정

ADC with DMA & DAC

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

학생 – 안상재
sangjae2015@naver.com

1. ADC with DMA

기존 방식

= AD 변환이 완료되면 ADC 결과를 하나씩 저장하는 코드를 작성

```
uint16_t ADCVal[4];
```

```
Waiting_ADConversion(CH1);  
ADCVal[0] = ADC->DR; // CH1 결과저장
```

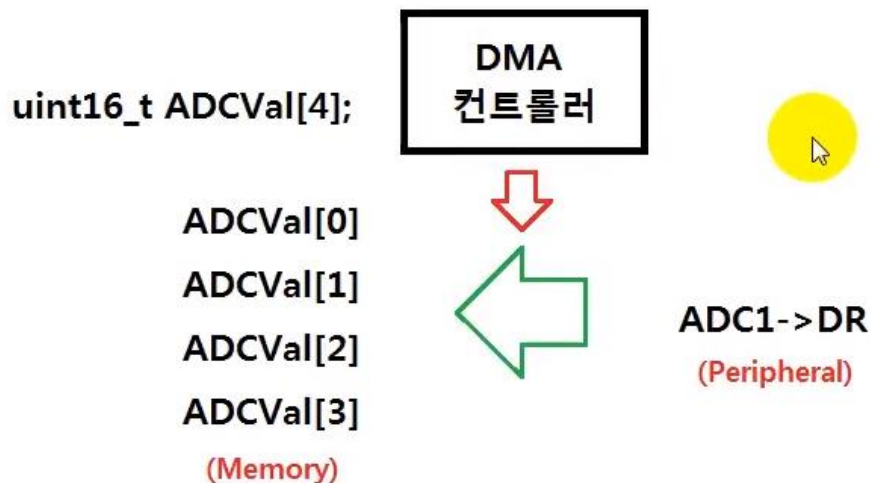
```
Waiting_ADConversion(CH2);  
ADCVal[1] = ADC->DR; // CH2 결과저장
```

```
Waiting_ADConversion(CH3);  
ADCVal[2] = ADC->DR; // CH3 결과저장
```

```
Waiting_ADConversion(CH4);  
ADCVal[3] = ADC->DR; // CH4 결과저장
```

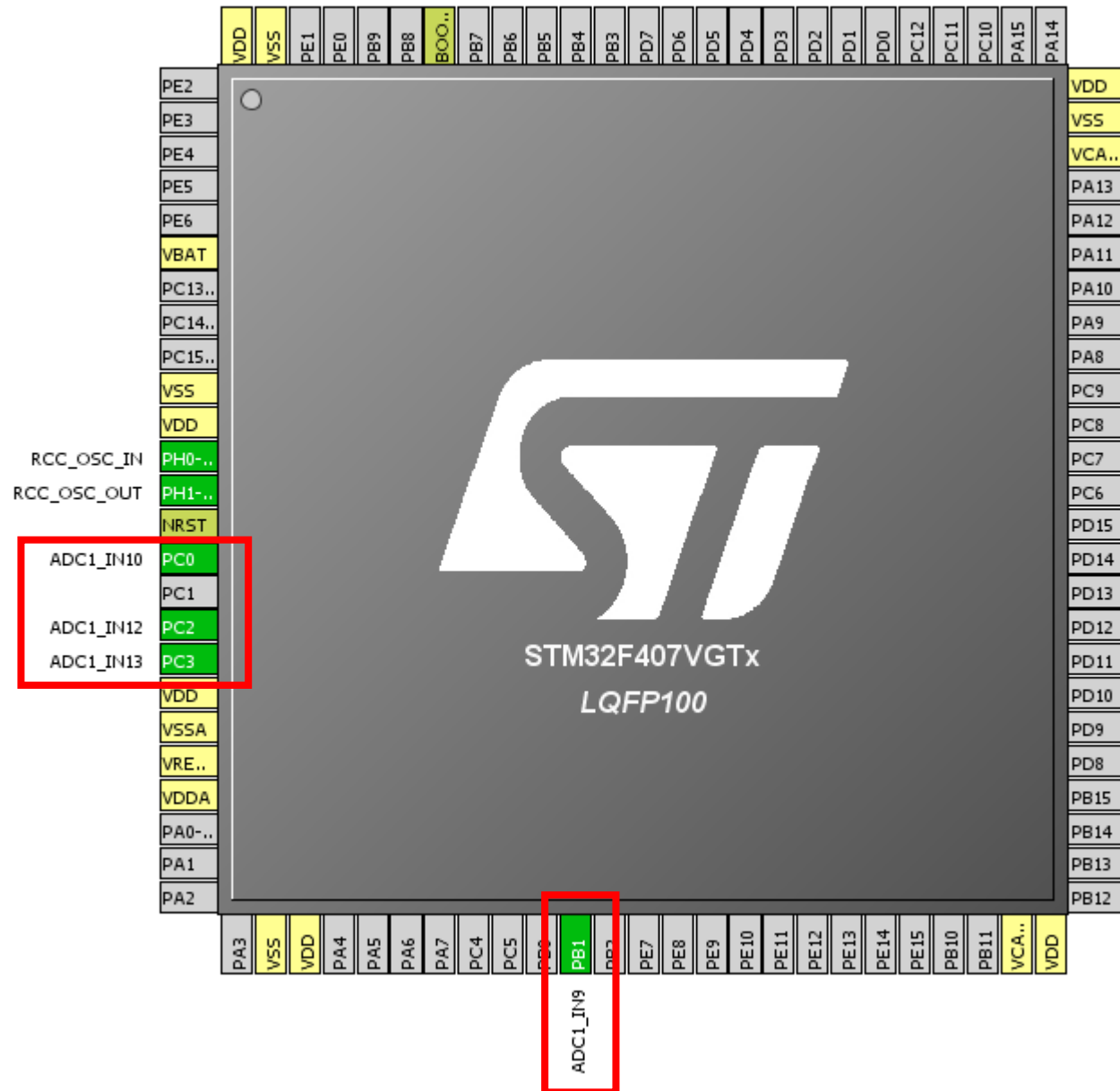
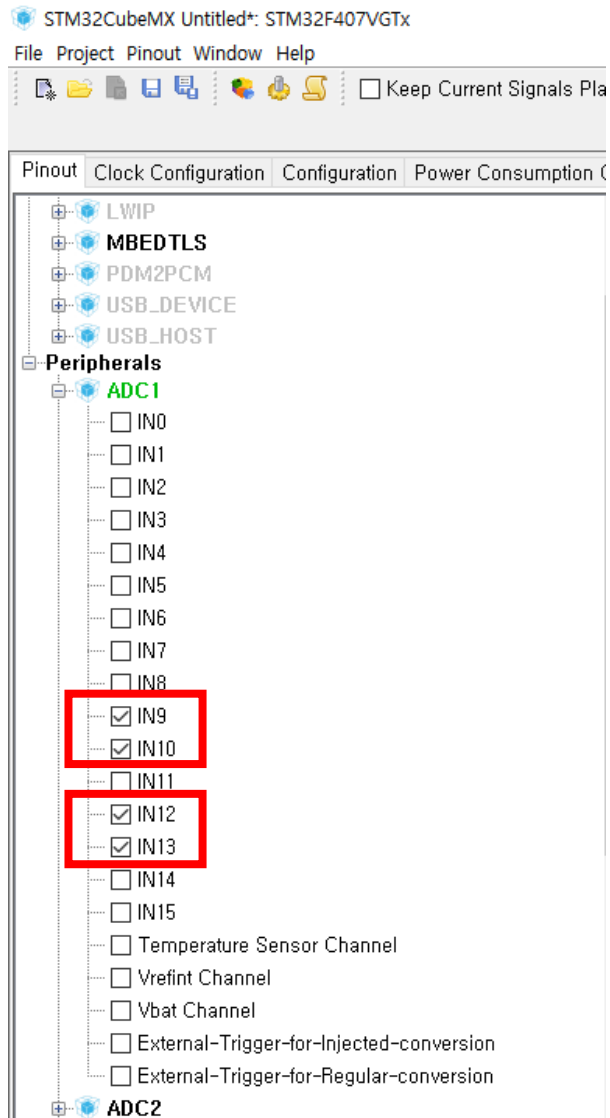
DMA 방식

= AD 변환이 완료되면 ADC 결과를 DMA 컨트롤러가 자동으로 원하는 변수에 저장 (Peripheral to Memory)

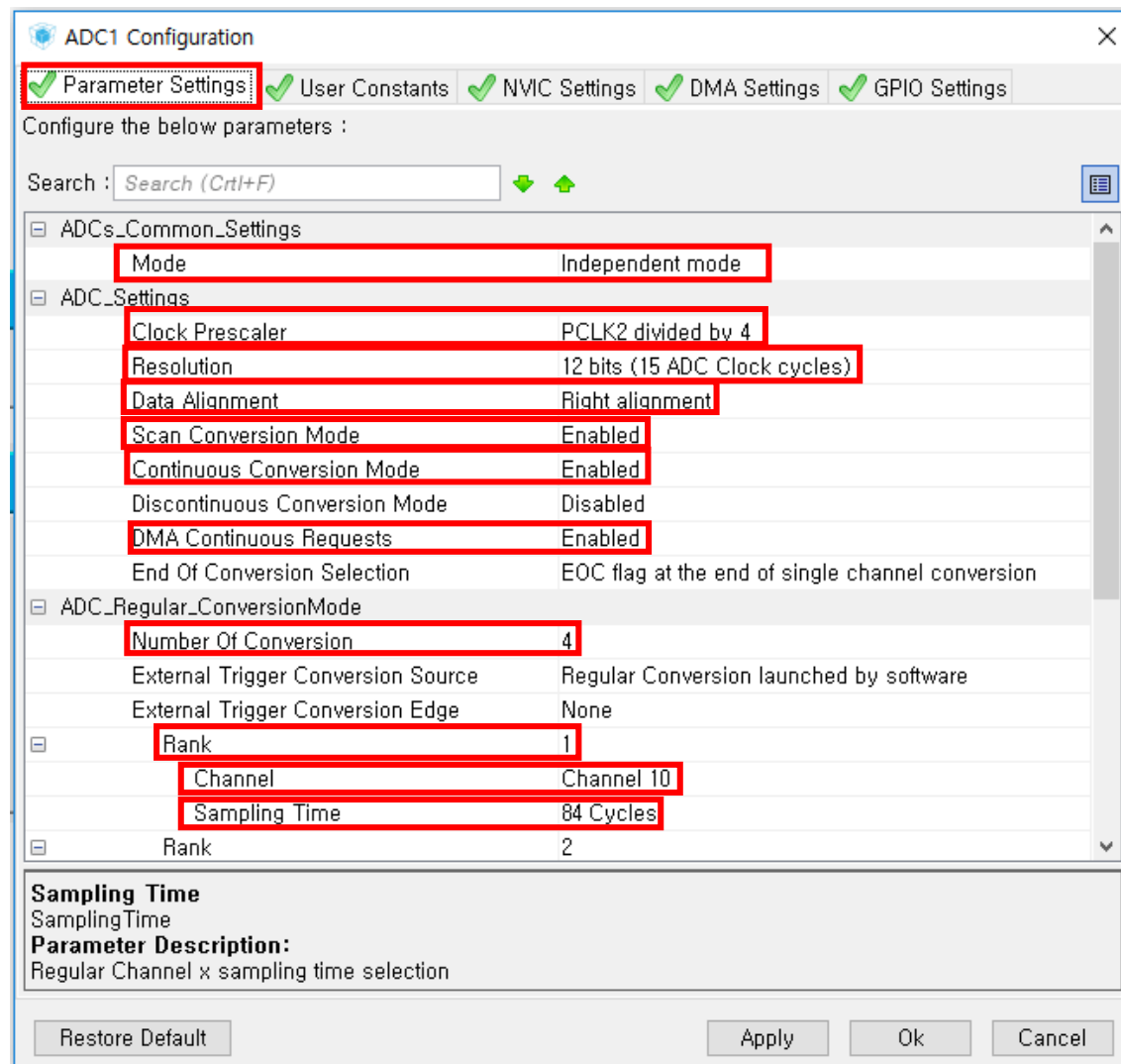
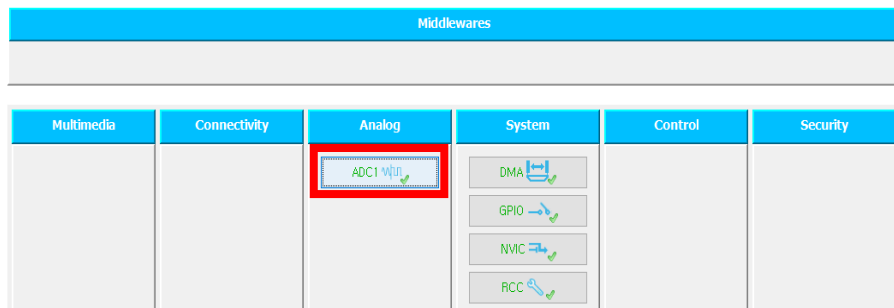


- 기존 방식은 ADC 변환 결과를 변수에 저장하는 코드를 통해 CPU가 클럭을 소비함
- DMA 방식은 DMA 컨트롤러가 자동으로 변수에 저장해주기 때문에 CPU가 클럭을 소비하지 않고, 그로 인해 CPU의 부담을 덜어줌!

- CubeMX 설정



Configuration 탭



Rank	2
Channel	Channel 12
Sampling Time	84 Cycles
Rank	3
Channel	Channel 13
Sampling Time	84 Cycles
Rank	4
Channel	Channel 9
Sampling Time	84 Cycles
ADC_Injected_ConversionMode	
Number Of Conversions	0
WatchDog	
Enable Analog WatchDog Mode	<input type="checkbox"/>

Sampling Time
SamplingTime
Parameter Description:
Regular Channel x sampling time selection

Restore Default Apply Ok Cancel

ADC 변환 순서가 ADCVal[0], ADCVal[1], ADCVal[2], ADCVal[3] 로 계속 반복됨

ADCVal[0]
ADCVal[1]
ADCVal[2]
ADCVal[3]
(Memory)



ADC1->DR
(Peripheral)

Peripheral 에서 Memory로 데이터 전송

ADC1 Configuration

Parameter Settings User Constants NVIC Settings **DMA Settings** GPIO Settings

DMA Request	Stream	Direction	Priority
ADC1	DMA2 Stream 0	Peripheral To Memory	Low

Peripheral(ADC1->DR) 의 주소는 바뀌면 안됨.
Memory(ADCVal[])에 저장을 하기 때문에 DMA 전송이 끝난 후,
Memory의 주소를 증가시켜야 함!

DMA Request Settings

Mode **Circular**

Increment Address ☐ ☒

Data Width **Half Word**

Burst Size

Restore Default Apply Ok Cancel

ADC resolution을 12bit로 했기 때문에 16bit인 Half Word로 설정

- 소스 코드

DMA를 통해 값이 변하는 변수는
volatile 을 쓰는 게 좋음!

```
int main(void)
{
    /* USER CODE BEGIN 1 */
    volatile uint16_t adcval[4]; // 최적화시키지 마라!
    uint8_t str[50];
    uint8_t i, rx;
    /* USER CODE END 1 */

    /* MCU Configuration-----*/

    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
    HAL_Init();

    /* USER CODE BEGIN Init */

    /* USER CODE END Init */

    /* Configure the system clock */
    SystemClock_Config();

    /* USER CODE BEGIN SysInit */

    /* USER CODE END SysInit */

    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_DMA_Init();
    MX_ADC1_Init();
    MX_USART2_UART_Init();

    /* USER CODE BEGIN 2 */
    HAL_ADC_Start_DMA(&hadc1, &adcval[0], 4);
```

초기화!

ADC DMA 시작!

adcval[0], adcval[1], adcval[2] 차례로 adc 결과가 들어감

터미널 창에서 'a' 를 전송하면 adc 결과 값을 출력함!

```
while (1)
{
/* USER CODE END WHILE */

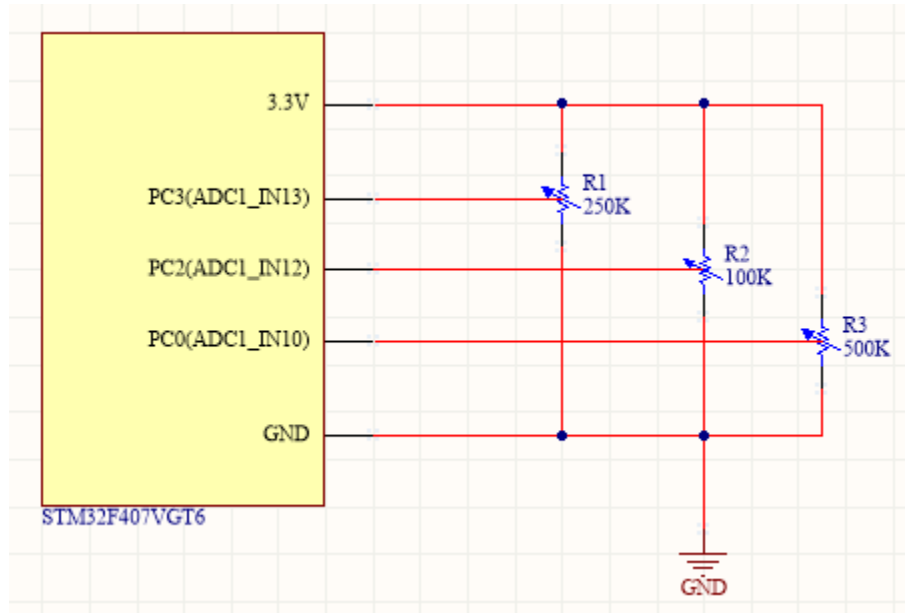
/* USER CODE BEGIN 3 */
    HAL_UART_Receive(&huart2, &rx, 1, 10);

    if(rx == 'a')
    {
        sprintf(str, "adcval[0] : %4d  adcval[1] : %4d  adcval[2] : %4d \r\n\n", adcval[0], adcval[1], adcval[2]);
        HAL_UART_Transmit(&huart2, str, sizeof(str), 10);

        for(i=0 ; i<100 ; i++)
            str[i] = 0;

        rx = 0;
    }
}
```

- adc 입력 회로도

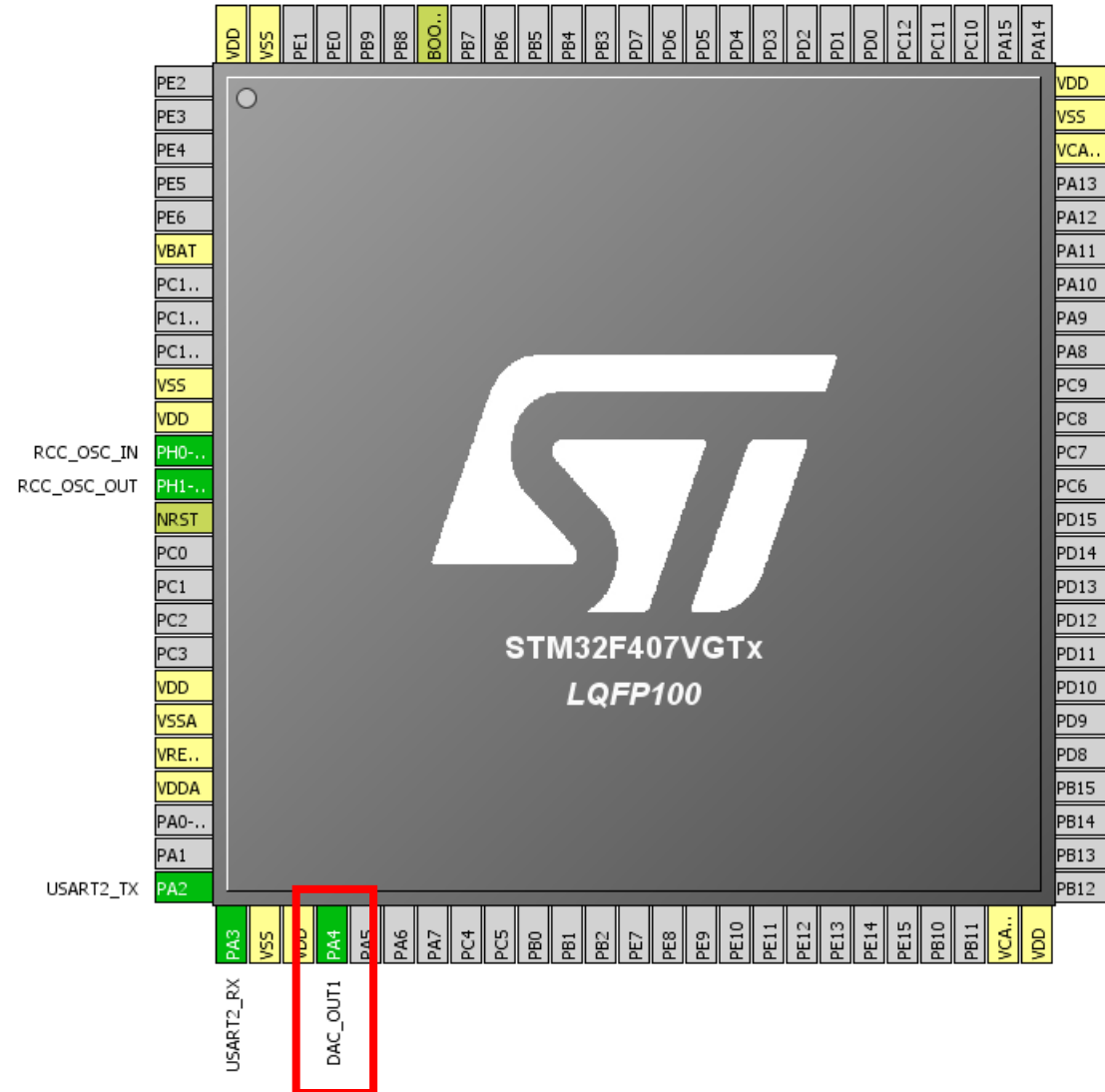
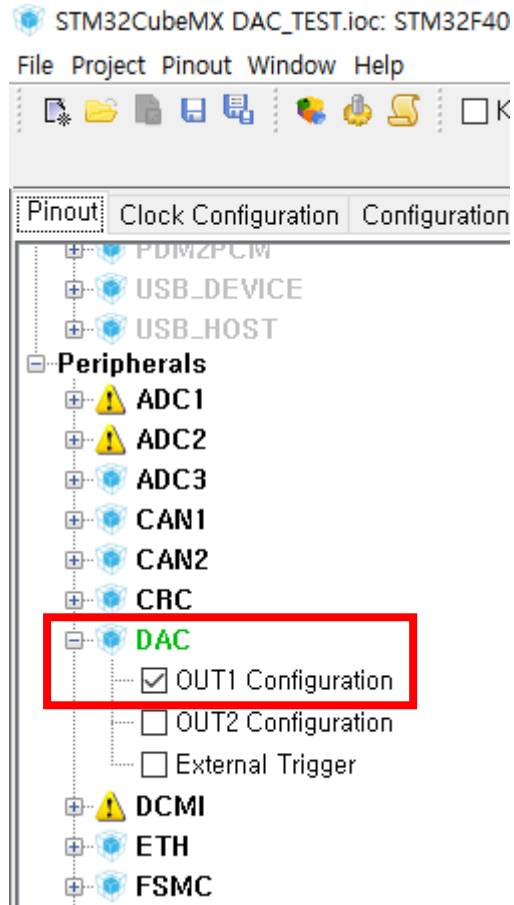


- 결과 화면 (가변저항을 돌리면서 결과값을 확인함)

```
문제점  태스크  콘솔  특성  Terminal
Serial COM19 (18. 10. 8 오전 12:07)
adcval[0] : 3722    adcval[1] : 4095    adcval[2] : 4095
adcval[0] : 3895    adcval[1] : 4095    adcval[2] : 4095
adcval[0] : 4016    adcval[1] : 4095    adcval[2] : 4095
adcval[0] : 4059    adcval[1] : 4095    adcval[2] : 4095
adcval[0] : 4095    adcval[1] : 4095    adcval[2] : 4095
adcval[0] : 4079    adcval[1] : 4095    adcval[2] : 4095
adcval[0] : 3949    adcval[1] : 4095    adcval[2] : 4095
adcval[0] : 3949    adcval[1] : 4095    adcval[2] : 4095
adcval[0] : 3951    adcval[1] : 4095    adcval[2] : 4095
adcval[0] : 3951    adcval[1] : 4095    adcval[2] : 4095
adcval[0] : 3927    adcval[1] : 4095    adcval[2] : 4095
adcval[0] : 3677    adcval[1] : 4095    adcval[2] : 4095
adcval[0] : 2      adcval[1] : 4095    adcval[2] : 4095
adcval[0] : 3      adcval[1] : 4095    adcval[2] : 4095
adcval[0] : 2      adcval[1] : 4085    adcval[2] : 4095
adcval[0] : 2      adcval[1] : 2954    adcval[2] : 4095
adcval[0] : 3      adcval[1] : 2951    adcval[2] : 4095
adcval[0] : 2      adcval[1] : 2950    adcval[2] : 4095
adcval[0] : 2      adcval[1] : 2950    adcval[2] : 3133
adcval[0] : 3      adcval[1] : 2949    adcval[2] : 2185
adcval[0] : 2      adcval[1] : 2953    adcval[2] : 1725
adcval[0] : 2      adcval[1] : 2961    adcval[2] : 1230
```



1. DAC

- CubeMX 설정



Middle

Analog

DAC 

DAC Configuration

Parameter Settings ✓ User Constants ✓ NVIC Settings ✓ DMA Settings ✓ GPIO Settings ✓

Configure the below parameters :

Search :

DAC Out1 Settings

Output Buffer	Enable
Trigger	None

Trigger

DAC_Trigger

Restore Default

Apply

Ok

Cancel

- 소스 코드

```
int main(void)
{
    /* USER CODE BEGIN 1 */

    /* USER CODE END 1 */

    /* MCU Configuration-----*/

    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
    HAL_Init();

    /* Configure the system clock */
    SystemClock_Config();

    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_DAC_Init();
    MX_TIM7_Init();
    MX_USART2_UART_Init();

    /* Initialize interrupts */
    MX_NVIC_Init();
    /* USER CODE BEGIN 2 */
    HAL_TIM_Base_Start_IT(&htim7);
    HAL_DAC_Start(&hdac, DAC_CHANNEL_1);
    /* USER CODE END 2 */
}
```

← DAC 초기화

← DAC 변환 시작

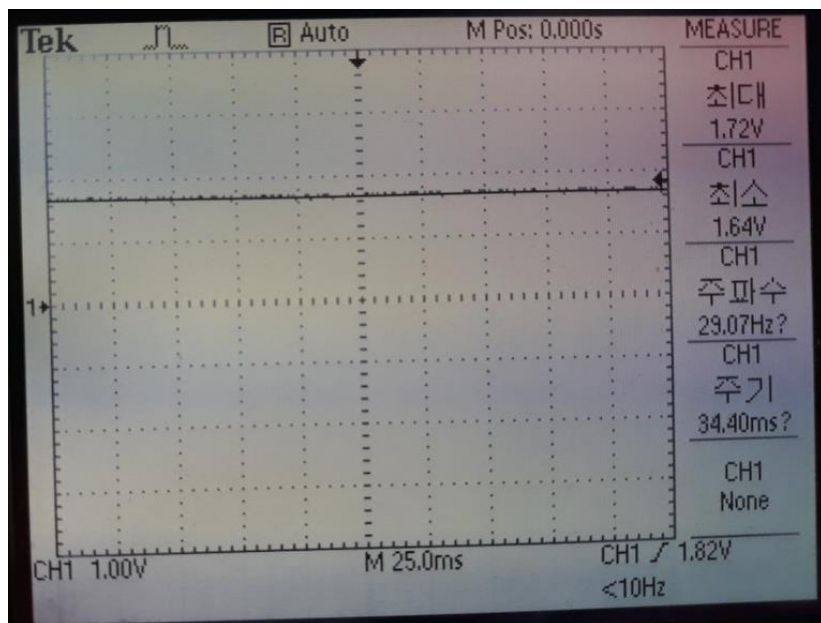
- 일정 전압 출력

```
while (1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, 2047);
    // dacval++;
    // if(dacval > 4095)
    //     dacval = 0;
}
```

- DAC_CHANNEL_1에서 변환
- DAC_ALIGN_12B_R : 12bit 오른쪽에 데이터를 배치함
- 0~4095까지 변환 가능 (4095 : 3V , 2047 : 1.65V)

- 결과 화면 (PA4 핀)



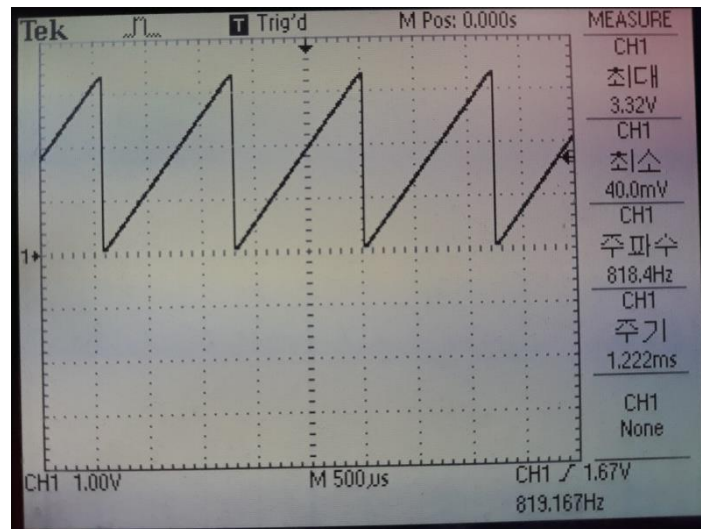
- 톱니파 출력

```
while (1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, dacval);

    dacval++;
    if(dacval > 4095)
        dacval = 0;
}
/* USER CODE END 3 */
```

- 결과 화면 (PA4 핀)



- sine함수 출력
- 주기 1ms TIMER 인터럽트 (main.c 에서 정의해 줌)

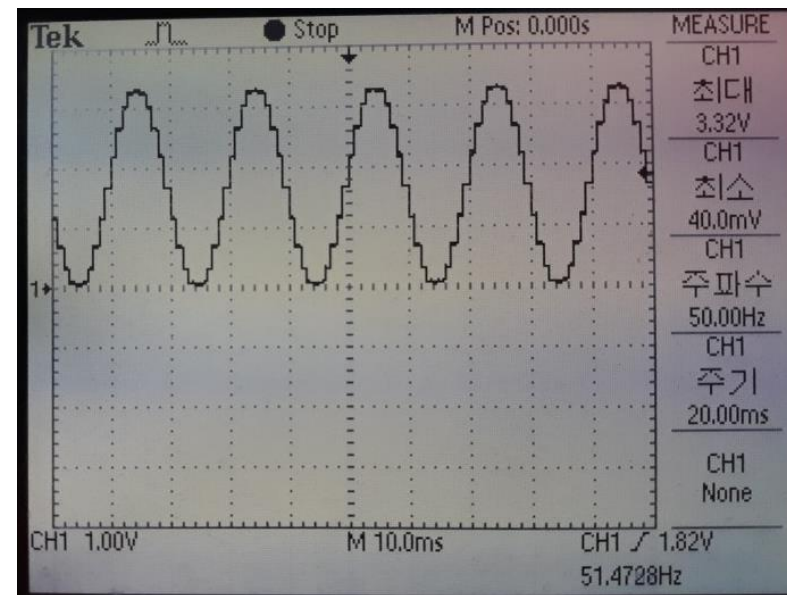
```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    static unsigned char cnt = 0;

    if(htim->Instance == TIM7)
    {
        HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, (sinf(2 * 3.14159265353 * 50 * cnt/1000) + 1) * 2047);
        cnt++;

        if(cnt > 999)
            cnt = 0;
    }
}
```

- 50HZ
- cnt / 1000으로 샘플링함
- sin함수의 범위는 -1~1 이고, 음수를 없애기 위해 1을 더함
- 최대 3.3V를 만들기 위해 2047을 곱함

- 결과 화면



- sine함수 출력 (샘플링 2배)

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    static unsigned char cnt = 0;

    if(htim->Instance == TIM7)
    {
        HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, (sinf(2 * 3.14159265353 * 50 * cnt/2000) + 1) * 2047);
        cnt++;

        if(cnt > 999)
            cnt = 0;
    }
}
```

- 결과 화면

