

TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

(eQEP 엔코더 속도측정)

강사 - 이상훈

gcccompil3r@gmail.com

학생 - 문한나

mhn97@naver.com

학생 - 장성환

redmk1025@gmail.com

eQEP

- 저속측정

34.2.2.4 eQEP Edge Capture Unit

The eQEP peripheral includes an integrated edge capture unit to measure the elapsed time between the unit position events as shown in [Figure 34-15](#). This feature is typically used for low speed measurement using the following equation:

$$v(k) = \frac{X}{t(k) - t(k - 1)} = \frac{X}{\Delta T} \quad (70)$$

where,

- X - Unit position is defined by integer multiple of quadrature edges (see [Figure 34-16](#))
- ΔT - Elapsed time between unit position events
- $v(k)$ - Velocity at time instant "k"

고정된 단위 시간 이벤트(거리)를 기준으로 두고, 단위 위치 이벤트 간의 경과 시간을 구하여 속도를 구하는 방법이다.

주의할 점은 위치 이벤트 사이의 주기 카운트 값이 65,535를 넘지 않아야 하며, 방향이 변경되어서는 안된다. 만약 오버플로우가 발생하거나 위치 이벤트 간 방향이 변경되면 상태 플래그에 오류 플래그가 설정된다. (QEPSTS [CDEF])

X : 고정된 단위 시간 이벤트

ΔT : 위치 이벤트 사이의 주기 카운트 값

$v(k)$: 시간 "k"에서의 속도

HCG 설정

TMS570LC4357ZWT PINMUX RTI GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1

General Driver Enable R5-MPU-PMU Interrupts VIM General VIM RAM VIM C

☒ Enable EQEP driver

☒ Enable EQEP1 driver **

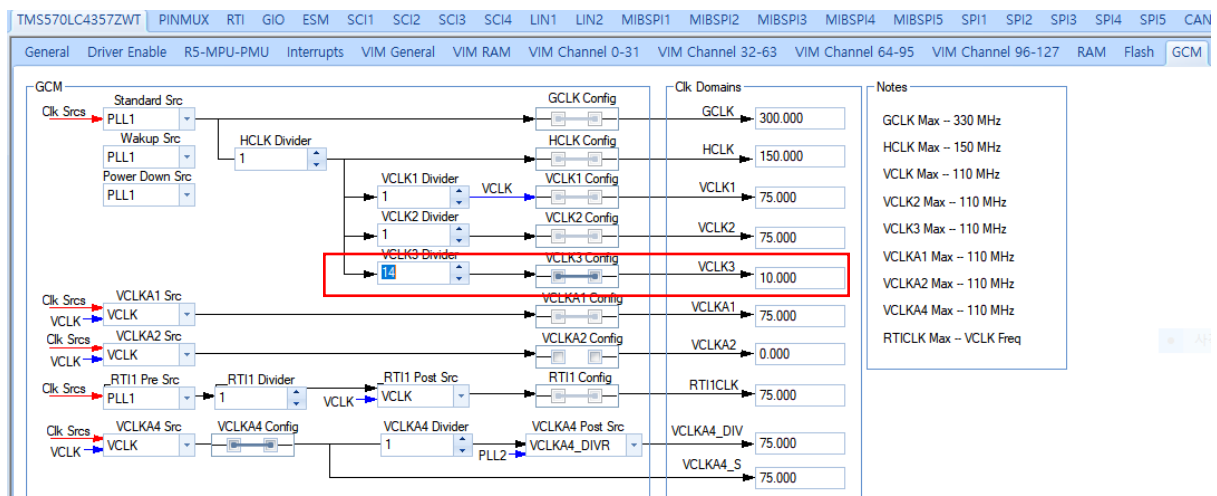
☐ Enable EQEP2 driver **

Start Page TMS570LC4357ZWT PINMUX RTI GIO ESM SCI1 SCI2 SCI3

Pin Muxing Input Pin Muxing Special Pin Muxing

Enable / Disable Peripherals

<input type="checkbox"/> HET1	<input type="checkbox"/> GIOA	<input type="checkbox"/> MIBSPI2	<input type="checkbox"/> MIBSPI1	<input type="checkbox"/> SCI3	<input type="checkbox"/> RMI
<input type="checkbox"/> HET2	<input type="checkbox"/> GIOB	<input type="checkbox"/> MIBSPI4	<input type="checkbox"/> MIBSPI3	<input type="checkbox"/> SCI4	<input type="checkbox"/> MII
<input type="checkbox"/> EMIF	<input checked="" type="checkbox"/> EQEP	<input type="checkbox"/> AD1EVT	<input type="checkbox"/> MIBSPI5	<input type="checkbox"/> LIN2/SCI2	<input type="checkbox"/> CAN4
<input type="checkbox"/> ETPWM	<input type="checkbox"/> ECAP	<input type="checkbox"/> AD2EVT	<input type="checkbox"/> I2C1	<input type="checkbox"/> I2C2	



EQEP1EQEP2

General Configuration

Position Counter Mode:

QUADRATURE_COUNT

Invert QEPxA Polarity

Invert QEPxB Polarity

Invert QEPxI Polarity

Invert QEPxS Polarity

Gate Index Pin with Strobe

Swap Quadrature Clock Input

External clock rate:

RESOLUTION_1x

Select QDIR:

CLOCKWISE

Compare Output Configurations

Sync Output Pin Select:

INDEX_PIN

Enable Sync Output

Enable Position Compare Shad:

Compare Value:

0x00000000

Sync Pulse Width:

0x000

x 4 VCLK4

Shadow Load Mode:

QPOSCNT_EQ_QPSCMP

Sync Output Polarity:

ACTIVE_HIGH

Position Counter Configuration

Counter Init Index Event:

RISING_EDGE

Max Position Count:

1x00000000

Counter Init Strobe Event:

DIRECTON_DEPENDENT

Init Counter on Index Event

Init Counter on Strobe Event

Enable SW Initialization

Position Counter Reset On:

MAX_POSITION

Init Position Count to:

0x00000000

Counter Latch Index Event:

RISING_EDGE

Counter Latch Strobe Event:

RISING_EDGE

Capture Configuration

Capture Timer Prescaler:

PS_8

Init Counter on Strobe Event

Unit Pos Event Prescaler:

PS_512

Unit Init Period:

0x00000000

Cap Timer Pos Mode:

ON_POSITION_COUNTER_READ

Interrupt Configuration

Position counter error Interrupt

Position-compare ready Interrupt

Quadrature phase error Interrupt

Position-compare match Interrupt

Quadrature direction change Interrupt

Strobe event latch Interrupt

Watchdog time out Interrupt

Index event latch Interrupt

Position counter underflow Interrupt

Unit time out Interrupt

Position counter overflow Interrupt

Watchdog Configuration

Watchdog Timer Value:

0x0000

CCS 코드

```
#include "HL_sys_common.h"
#include "HL_eqep.h"
#include "HL_sys_core.h"
#include "stdio.h"

#define UNIT_POSITION_X 106.667 //각도

void main(void)
{
    uint32 deltaT = 0U;
    double velocity = 0U;

    double tmp = 0U;

    /* EQEP initialization based on GUI Configuration. */
    QEPInit();

    /* Enable Position Counter */
    eqepEnableCounter(eqepREG1);

    /* Enable Unit Timer. */
    eqepEnableUnitTimer(eqepREG1);

    /* Enable capture timer and capture period latch. */
    eqepEnableCapture(eqepREG1);

    while(1)
    {
        /* Status flag is set to indicate that a new value is latched in the QCPRD
        register. */
        if((eqepREG1->QEPSTS & 0x80U) !=0U)
        {
            /* Elapsed time between unit position events */
            deltaT = eqepREG1->QCPRD;

            printf("deltaT:%d\n", deltaT);

            tmp = (double)((double)deltaT / (1000000.0 / 8.0));

            printf("tmp : %f\n", tmp);

            /* Calculate Velocity from deltaT and the value of the unit position. */
            /* The value of Unit Position is a sample value and should be changed
            by the User as per the actual value in the UNIT_POSITION_X macro above. */
            velocity = (double)(UNIT_POSITION_X/tmp);
        }
    }
}
```

```
        /* Clear the Status flag. */  
        eqepREG1->QEPSTS |= 0x80U;  
  
        printf("velocity:%f\n", velocity);  
    }  
}  
}
```

결과

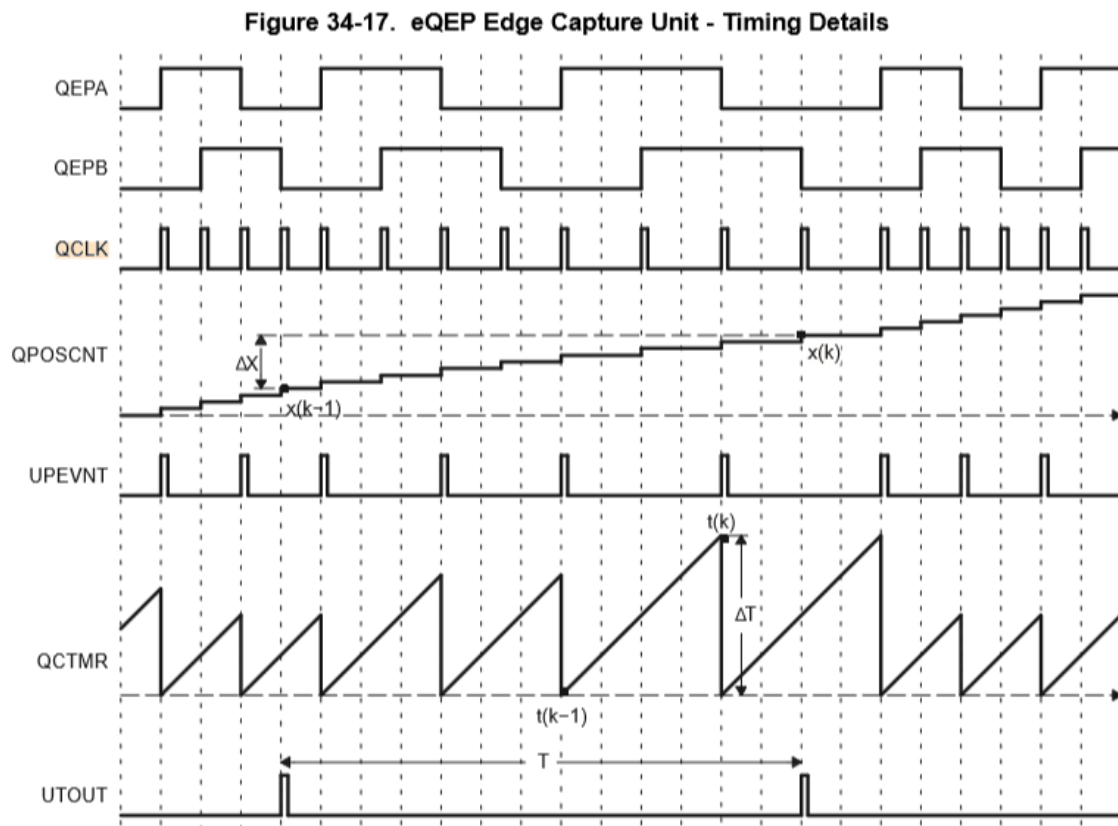
Console

eqep:CIO

```
deltaT:39164  
tmp : 0.031331  
velocity:3404.497753  
deltaT:39243  
tmp : 0.031394  
velocity:3397.644166  
deltaT:39626  
tmp : 0.031701  
velocity:3364.804674  
deltaT:39898  
tmp : 0.031918  
velocity:3341.865507  
deltaT:39515
```

코드설명

UNIT_POSITION_X 106.667



모터의 분해능이 432P/R이므로 한바퀴 돌 때 마다 파형이 432개가 나온다.

모터 사양 MOTOR SPECIFICATION

항목	사양	항목	사양
정격 회전수	2,000 rpm (오차범위 10%)	정격 전류	1.3A이하
정격 출력	12W	무부하 전류	0.4A이하
정격 토크	0.6 Kg-cm	무부하 회전수	2,800rpm
엔코더 전원	DC 5v (오차5%)	방식	인크리멘탈
분해능	432 P/R	출력 파형	단형파
출력 전압	L=0.4v/10mA 이하	응답 주파수	36Khz이하
제품 중량	500g 이하	제품 소음	50cm에서 54db이하

Quadrature Encoder는 한 주기마다 4번의 신호가 나오므로 엔코더의 분해능은 $432 * 4 = 1728$ 이 된다. (QCLK)

HCG에서 Unit position event prescaler를 512로 맞춰주었기 때문에 거리(각도)는 $1728 / 512 = 106.667$ 로 설정했다.

Capture Configuration

Capture Timer Prescaler: PS_8

☐ Init Counter on Strobe Event

Unit Pos Event Prescaler: PS_512
Unit Init Period: 0x00000000

Cap Timer Pos Mode: ON_POSITION_COUNTER_READ

deltaT = eqepREG1->QCPRD;

34.3.23 eQEP Capture Period Register (QCPRD)

Figure 34-43. eQEP Capture Period Register (QCPRD) [offset = 3Eh]



LEGEND: R/W = Read/Write; -n = value after reset

Table 34-26. eQEP Capture Period Register (QCPRD) Field Descriptions

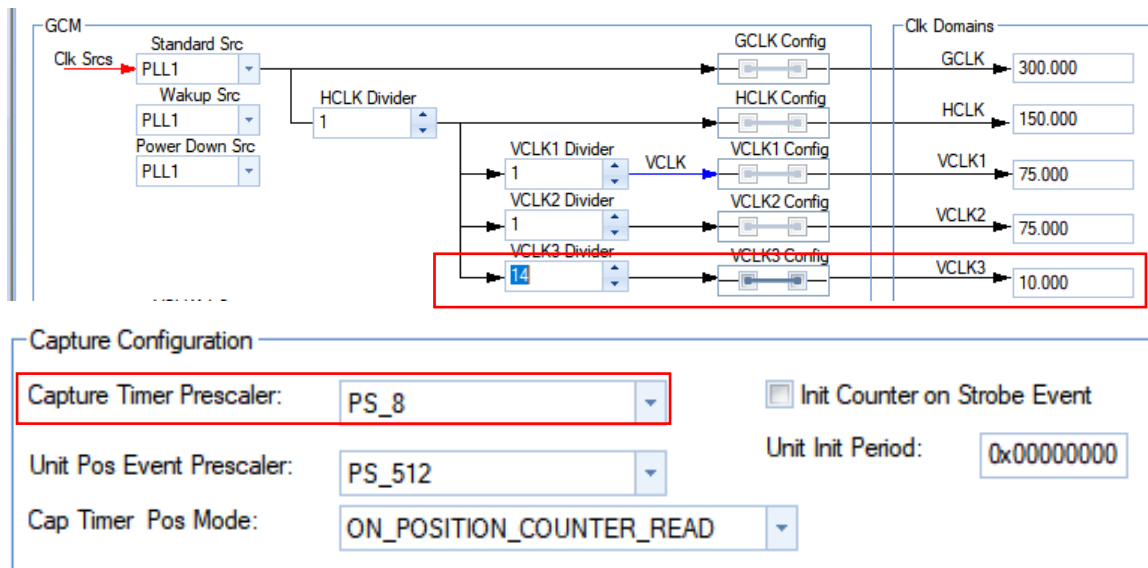
Bits	Name	Description
15-0	QCPRD	This register holds the period count value between the last successive eQEP position events.

QCPRD는 위치이벤트 사이의 주기 카운트값이다.

QCTMR(eQEP캡처 타이머)이 VCLK3의 주파수를 기반으로 계속 카운트하고, UPEVNT를 만나면 초기화 된다. 카운트된 값은 **QCPRD**(위치이벤트 사이의 주기 카운트값)에 저장된다.

```
tmp = (double)((double)deltaT / (10000000.0 / 8.0));
```

카운트한 값에서 주파수(10MHz)와 프리스케일한 8을 또 나눠준다.



```
velocity = (double)(UNIT_POSITION_X/tmp);
```

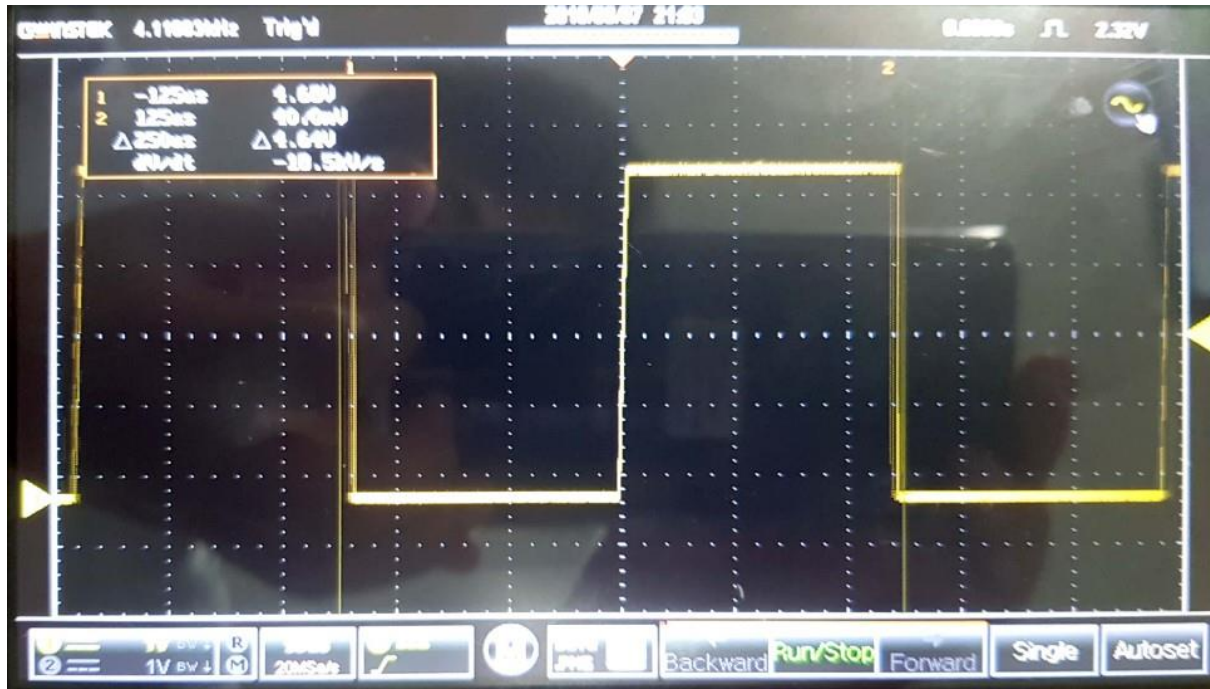
마지막으로 각속도를 구하기 위해 아까 설정한 106도(각도)에서 구한 시간을 나눠준다.

$$\text{Velocity} = \frac{106.67}{\text{QCPRD} / (10\text{MHz} / 8)}$$

따라서 결과를 보면 약 3400정도가 나온다.

$3400 / 360 = 9.4$ 로 1초에 9.4바퀴 도는 것을 알 수 있다.

$$\text{회전수} = \frac{4110}{432} = 9.51$$



실제 측정값과 거의 동일한 수치를 나타낼 수 있다.

문제점은 2800rpm 이상이면 제대로 된 결과값이 나오지 않는다.

eQEP를 사용한 고속 속도 측정

큰 모터 속도와 높은 센서 분해능의 조합은 시간 간격 ΔT 를 작게 만들어 타이머 분해능에 더 큰 영향을 받는다. 속도 범위가 큰 시스템의 경우 (즉, 저속 및 고속 둘다 속도 추정이 필요한 경우) 저속 측정 방법 뿐만 아니라 고속 측정법이 필요하다.

고속 측정법은 다음과 같은 방정식을 사용한다.

$$V(k) = \frac{X(k) - X(k-1)}{T} = \frac{\Delta X}{T}$$

$V(k)$: 어떠한 k 인 순간의 속력.

$X(k) - X(k-1)$: 고정된 시간 T 가 지난 최종 위치에서 처음 위치를 뺀 거리 (이동 거리)

T : 고정된 시간 (매 측정하고 싶은 시간을 설정)

즉, 모터를 T마다 이동거리를 측정하여 $\frac{\text{거리}}{\text{시간}} = \text{속력}$ 을 이용하여 매 T 초마다의 순간 속력을 얻는다.

General Configuration	
Position Counter Mode:	<div>QUADRATURE_COUNT</div> <div> <input type="checkbox"/> Invert QEPxA Polarity <input type="checkbox"/> Invert QEPxB Polarity <input type="checkbox"/> Invert QEPxI Polarity <input type="checkbox"/> Invert QEPxS Polarity <input type="checkbox"/> Gate Index Pin with Strobe <input type="checkbox"/> Swap Quadrature Clock Input </div>
External clock rate:	<div>RESOLUTION_1x</div>
Select QDIR:	<div>CLOCKWISE</div>

Position Counter Configuration	
Counter Init Index Event:	<div>RISING_EDGE</div> <div>Max Position Count: 0x00FFFFFF</div>
Counter Init Strobe Event:	<div>DIRECTON_DEPENDENT1</div> <div> <input type="checkbox"/> Init Counter on Index Event <input type="checkbox"/> Init Counter on Strobe Event <input type="checkbox"/> Enable SW Initialization </div>
Position Counter Reset On:	<div>MAX_POSITION</div>
Counter Latch Index Event:	<div>RISING_EDGE</div>
Counter Latch Strobe Event:	<div>RISING_EDGE</div> <div>Init Position Count to 0x00000000</div>

Capture Configuration	
Capture Timer Prescaler:	<div>PS_4</div> <div> <input type="checkbox"/> Init Counter on Strobe Event </div>
Unit Pos Event Prescaler:	<div>PS_512</div> <div>Unit Init Period: 0x00000200</div>
Cap Timer Pos Mode:	<div>ON_UNIT_TIMEOUT_EVENT</div>

Compare Output Configurations

Sync Output Pin Select: INDEX_PIN

☐ Enable Sync Output

Shadow Load Mode: QPOSCNT_EQ_QPSCMP

☐ Enable Position Compare Shadc

Sync Output Polarity: ACTIVE_HIGH

Compare Value: 0x00000000

Sync Pulse Width: 0x000 x 4 VCLK4

Interrupt Configuration

☐ Position counter error Interrupt

☐ Position-compare ready Interrupt

☐ Quadrature phase error Interrupt

☐ Position-compare match Interrupt

☐ Quadrature direction change Interrupt

☐ Strobe event latch Interrupt

☐ Watchdog time out Interrupt

☐ Index event latch Interrupt

☐ Position counter underflow Interrupt

☐ Unit time out interrupt

☐ Position counter overflow Interrupt

Watchdog Configuration

Watchdog Timer Value: 0x0000

VCLK3를 분주하여 15MHz로 맞춰주고 PINMUX 설정을 완료한다.

사용한 모터의 엔코더의 스펙은 다음과 같다.

432P/R으로 모터가 1초에 1 회전할 때, A상과 B상이 90도의 위상차를 가지고 432Hz로 동작한다.

eQEP QUADRATURE_COUNT 모드의 특성 상, $432\text{ Hz} \times 4$ 를 통하여 1728Hz로 동작하게 된다.

다음으로 eQEP 고속 측정 모드의 코드를 확인하면,

```

#include <HL_eqep.h>
#include <HL_hal_stdtypes.h>
#include "HL_sys_common.h"
#include "HL_system.h"
#include "HL_can.h"
#include "HL_esm.h"
#include "HL_sys_core.h"
#include "stdio.h"

#define UNIT_TIME_T 0.1 //거리
#define prescaler 1875000.0 // 분주비
#define encoder_trans_degree_coefficient 0.83333 //
#define circle_angle 360.0 // 1회전 각도

int main(void)
{
    uint32 distance_cnt = 0;
    //double angle_distance[2] = {0,0};
    //uint8 distance_flag = 1;
    uint32 dx = 0;
    double angle_velocity = 0;

    QEPIInit();
    /* Enable Position Counter */
    eqepEnableCounter(eqepREG1);
    /* Enable Unit Timer. */
    eqepEnableUnitTimer(eqepREG1);
    /* Enable capture timer and capture period latch. */
    eqepEnableCapture(eqepREG1);

    eqepREG1->QEPCNTL |= 0x3002; //Position counter reset mode = 0x11 (unit time
reset mode), eQEP unit timer enable.
    eqepREG1->QCAPCTL |= 1 << 15; // Enable eQEP capture.
    //eqepREG1->QUPRD = 0x2DC6C; // Time set is 187500 count -> GCM = 15MHz,
prescaler = 8 so, capture time = 1875000hz and dt = 0.1(fixed value)
    eqepREG1->QUPRD = 0x5B8D8; // Time set is 375000 count -> GCM = 15MHz,
prescaler = 4 so, capture time = 3750000hz and dt = 0.1(fixed value)

    while (1)
    {
        distance_cnt = (eqepREG1->QPOSLAT);
        angle_velocity = (double)(distance_cnt*encoder_trans_degree_coefficient) /
UNIT_TIME_T;

        printf("각 속도: %f\n", angle_velocity);
        printf("rpm : %f \n", (angle_velocity / circle_angle)*60);
        printf("초당 회전 수: %f\n", angle_velocity/circle_angle);

    }
    return 0;
}

```

각각의 계수를 확인하여 보자.

```
#define UNIT_TIME_T 0.1 //거리
```

해당 계수는 설계자가 원하는 시간을 입력하는 것이다.

```
eqepREG1->QUPRD = 0x5B8D8;
```

15Mhz의 분주비를 4로 나누면, 3750000hz의 오실레이터가 된다. 해당 오실레이터가 375000번이 동작하면, 0.1s의 주기를 갖게 된다.

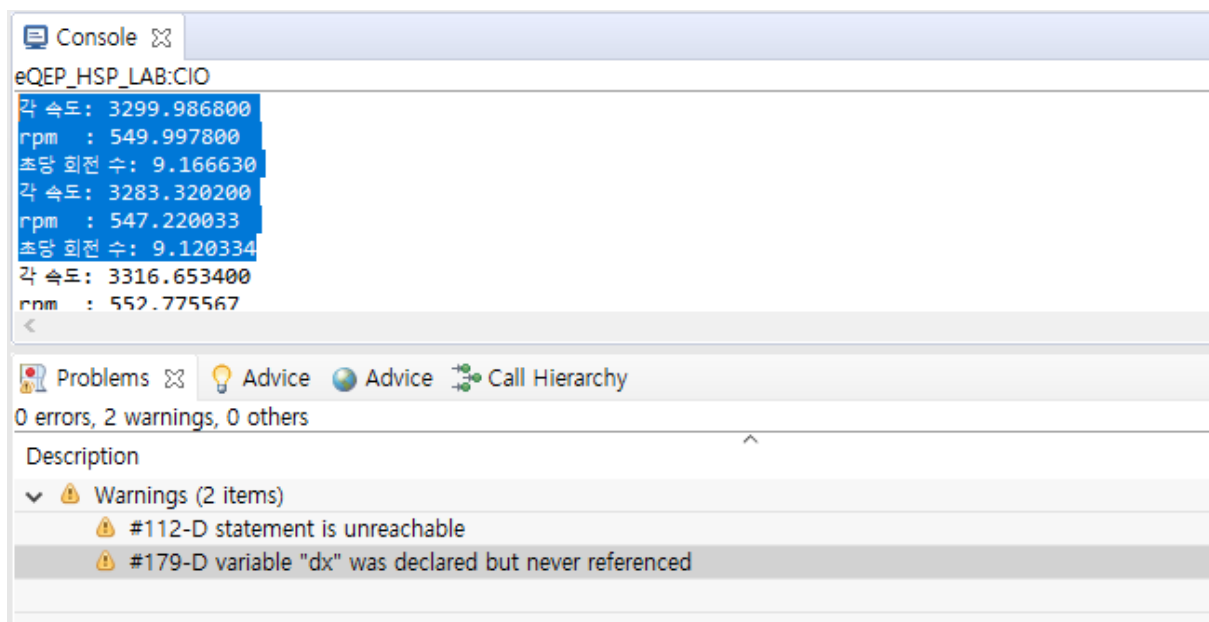
```
#define encoder_trans_degree_coefficient 0.83333 //
```

엔코더의 스펙은 432P/R 이므로, 360(degree) / 432 를 하면 0.83333이 된다.

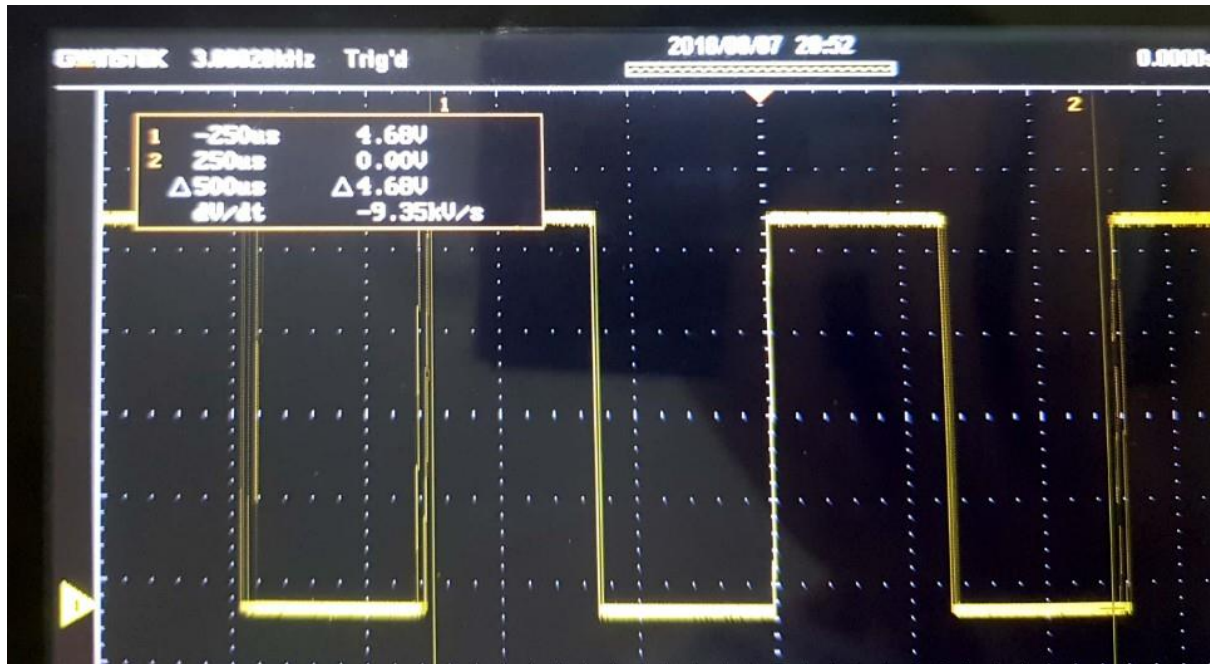
해당 레지스터는 degree나 radian을 반환하는 것이 아니라 count의 값을 반환하므로, count를 이용하여 degree나 radian으로 변환해야 한다.

결과를 확인하면 다음과 같다.

해당 결과값은 5볼트의 입력을 모터에 인가하였을 때, 속력을 나타낸다.



결과를 검증하기 위하여 오실로스코프로 파형을 확인한다.



사진상으로는 조금 확인하기 힘들지만, 주파수가 3.9kHz 근처를 나타낸다.

$$\text{회전수} = \frac{3900}{432} = 9.02 \text{ 회/전}$$

거의 동일한 수치를 나타냄을 알 수 있다.

저속모드에서 2800rpm 이상인 경우에 문제점이 발생하였는데, 해당 고속모드로는 정확한 결과값이 출력되었다.

저속모드로 측정을 하다가 입력 받은 속도 값이 어느 임계치 이상이면 고속모드로 전환하여 측정해야 모터의 출력을 정확히 측정 할 수 있다.