03 진행상황 및 문제점

김시윤 – MCU eCAP 개념 요약 및 실험

eCAP 개념

33.1 Introduction

Uses for eCAP include (eCAP용도는 다음과 같다):

• Speed measurements of rotating machinery (for example, toothed sprockets sensed via Hall sensors)
- 회전 기계의 속도 측정 (예 , 홀 센서를 통해 감지 된 톱니 형 스포로킷)

• Elapsed time measurements between position sensor pulses
- 위치 센서 펄스 간의 경과 시간 측정

• Period and duty cycle measurements of pulse train signals
- 펄스 트레인 신호의주기 및 듀티 사이클 측정

• Decoding current or voltage amplitude derived from duty cycle encoded current/voltage sensors
- 듀티 사이클 인코딩(부호화) 된 전류 / 전압 센서에서 파생 된 전류 또는 전압 진폭 디코딩(복호화)

eCAP 개념

33.1.1 Features

eCAP 모듈에는 다음과 같은 기능이 있습니다.

- 4개의 이벤트 타임 스탬프 레지스터 (4-event time-stamp registers) 각 32비트

- 4개의 타임 스탬프 캡쳐 이벤트 시퀀스에 대한 에지극성 선택. CEVT1~4 를 발생할 에지 선택 가능

- 네 가지 이벤트 중 하나에서 인터럽트

- 최대 4개의 이벤트 타임 스탬프 만큼 단일 샷 캡쳐 .

- 4 깊이 원형 버퍼에서(CAPx) 타임 스탬프의 연속 모드 캡처

- 절대(절대시간) 타인스탬프 켭쳐

– Difference (Delta) mode 타임스탬프 캡쳐

- 단일 입력 핀 전용 리소스.

-캡처 모드에서 사용하지 않으면 ECAP 모듈을 단일 채널 PWM 출력으로 구성 할 수 있습니다

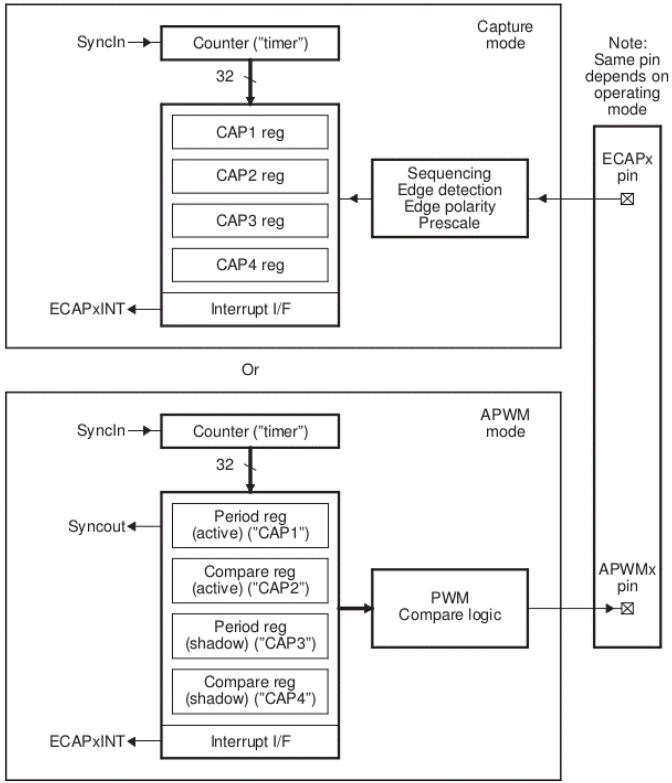
eCAP 개념 - APWM

Capture Mode 를 사용하지 않을 때 eCAP 핀으로 PWM을 생성 할 수 있다.
카운터인 TSCTR 은 count up mode로 작동하여 시간(주기)를 측정한다.
CAP1 및 CAP2 레지스터는 활성 기간 및 비교 레지스터가되고
CAP3 및 CAP4 레지스터는 각각주기 및 캡처 섀도 레지스터가 됩니다.

A. eCAP 핀은 Capture 모드와 APWM 모드를 지원합니다. Capture 모드에서는 이 핀은 입력핀이 되고, APWM모드에서는 이 핀은 출력핀이 됩니다.

B. APWM 모드에서 CAP1/CAP2 값을 쓰면 같은 값이 Shadow 레지스터인 CAP3/CAP4 에도 들어갑니다. Shadow 레지스터를 쓸라면 Shadow 모드를 호출해야 합니다.

Figure 33-1. Capture and APWM Modes of Operation



A A single pin is shared between CAP and APWM functions. In capture mode, it is an input; in APWM mode, it is an output.

B In APWM mode, writing any value to CAP1/CAP2 active registers also writes the same value to the corresponding shadow registers CAP3/CAP4. This emulates immediate mode. Writing to the shadow registers CAP3/CAP4 invokes the shadow mode.

eCAP 개념 - Continuous/One-Shot Control

Mod4 (2 비트) 카운터는 에지 규정 이벤트 (CEVT1-CEVT4)를 통해 증가합니다.

Mod4 카운터는 카운팅을 계속하고 (0-> 1-> 2-> 3-> 0) 멈추지 않으면 wraps around 합니다.

2비트 stop register는 Mod4 카운터 출력과 비교되고 Mod4 카운터 값과 같을 때 CAP1-CAP4 레지스터 의 로드를 금지시킵니다. 이 동작은 원샷모드일때 발생한다.

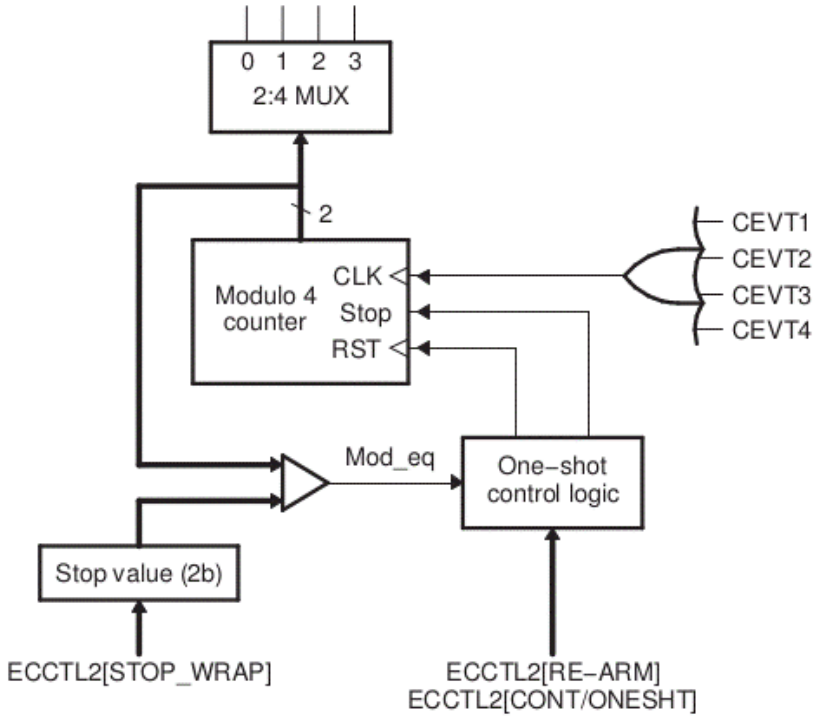
continuous/one-shot block은 stop-value comparator 에 의해 작동되고 소프트웨어 제어를 통해 다시 활성화 될 수 있는 모노 샷 타입의 동작을 통해 Mod4 카운터의 시작/정지 및 리셋(0) 기능을 제어한다.

일단 동작하면, eCAP 모듈은 Mod4 카운터와 CAP1~4 레지스터(time-stamps)를 고정하기 전에 capture events 1-4 (CEVT1-4, defined by stop-value) 를 기다린다.

재 활성화는 다른 캡처 시퀀스를 위해 eCAP 모듈을 준비합니다. 또한 다시 활성화하면 Mod4 카운터가 지워지고 CAPLDEN(CAPx 를 로드할지 말지 결정하는 레지스터) 비트가 1로 설정되면 CAP1-4 레지스터를 다시 로드 할 수 있습니다.

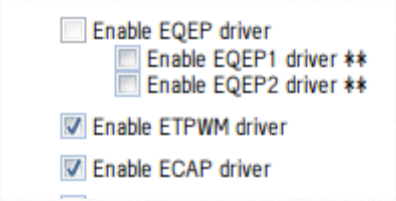
연속 모드에서 Mod4 카운터는 계속 실행됩니다 (0-> 1-> 2-> 3-> 0, 원샷 동작은 무시되며 캡처 값은 원형 버퍼 시퀀스인 CAP1-4에 계속 기록됩니다.

Figure 33-5. Continuous/One-shot Block

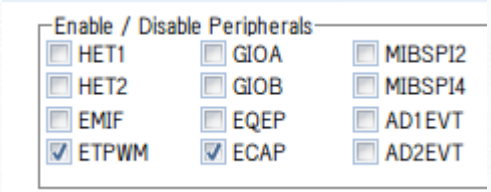


HalCogen setting

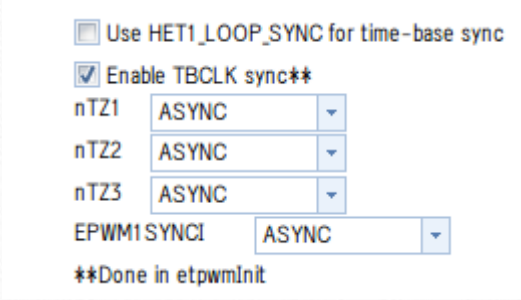
Driver enable



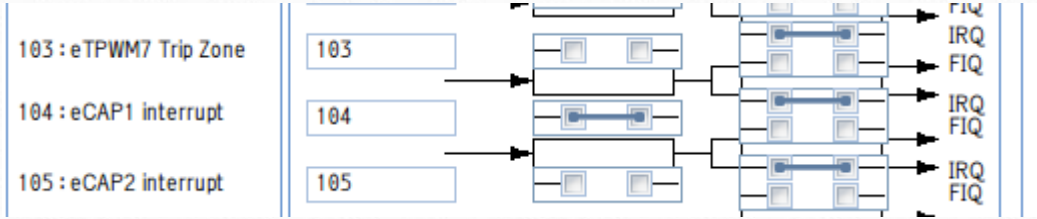
Pin Mux



Special pin Mux



Vm channel

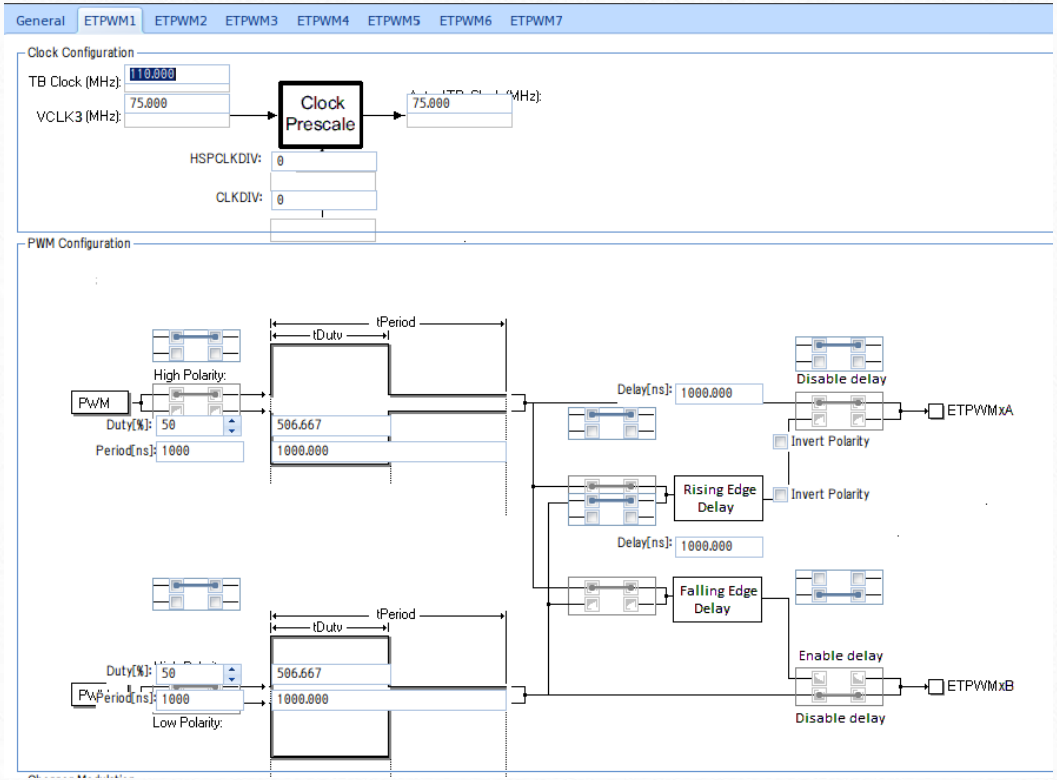


HalCogen setting

ETPWM Setting

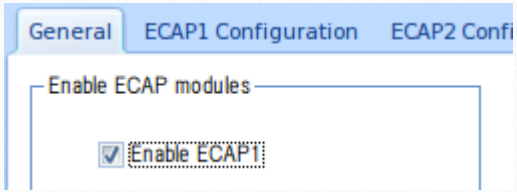
Prd = 1000 ns

Duty = 50%

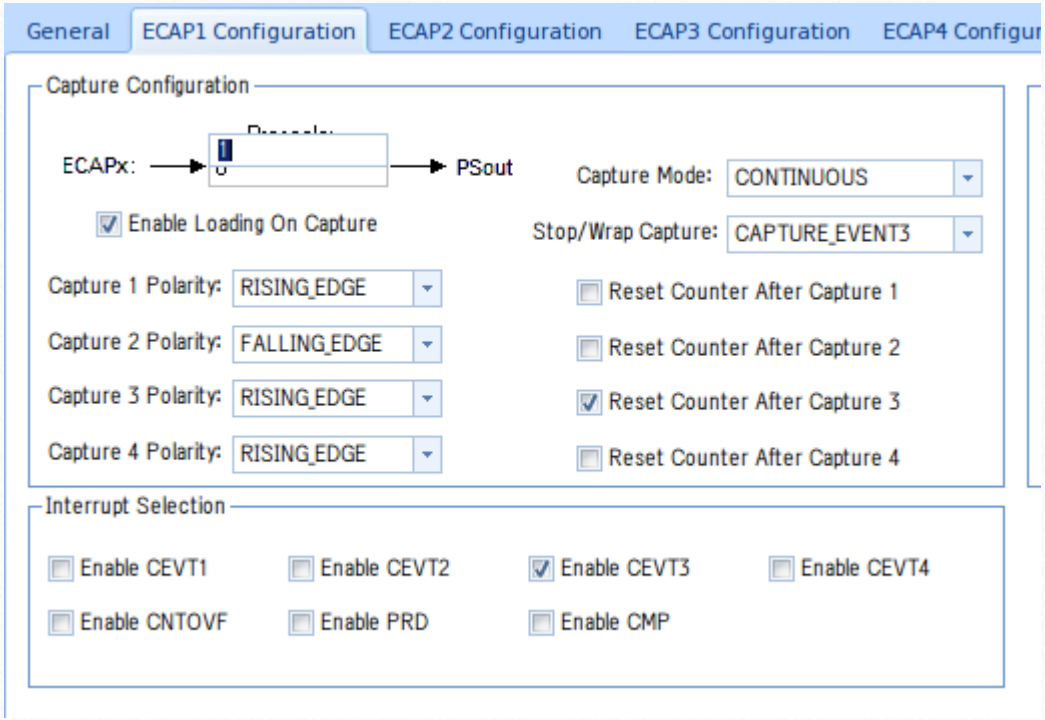


HalCogen setting

ECAP General



ECAP1 Configuration



CONTINUOUS MODE

CEVT3 을 stop/wrap value 로 설정
Mod 4 카운터가 stop/wrap 값과 같아질 경우 CAPx 레지스터를 초기화

CEVT3 인터럽트 활성화
-> CAP3이 캡쳐하면 인터럽트 발생.

CCS code

```
#include "HL_sys_common.h"
#include "stdio.h"
#include "HL_system.h"
#include "HL_etpwm.h"
#include "HL_ecap.h"


void main(void)
{
    _enable_interrupt_();

    etpwmInit();
    ecapInit();
    etpwmStartTBCLK();
    ecapStartCounter(ecapREG1);
    ecapEnableCapture(ecapREG1);

    while(1);
}
```

```
void ecapNotification(ecapBASE_t *ecap,uint16 flags)
{
    uint32 cap1, cap2, cap3;
    float64 duty, period;

    cap1 = ecapGetCAP1(ecapREG1);
    cap2 = ecapGetCAP2(ecapREG1);
    cap3 = ecapGetCAP3(ecapREG1);

    duty = (cap2 - cap1)*1000/VCLK3_FREQ;
    period = (cap3 - cap1)*1000/VCLK3_FREQ;

    printf("cap1 = %d₩n",cap1);
    printf("cap2 = %d₩n",cap2);
    printf("cap3 = %d₩n",cap3);
    printf("Duty = %fns₩n", duty);
    printf("Period = %fns₩n₩n", period);

}
```

출력 결과

```
[CortexR5]          Duty              =          57265580.000000ns
Period                    =                    57266072.000000ns

cap1                              =                          74
cap2                              =                          25
cap3                              =                          62
Duty                         =                    506.666656ns
Period                        =                    1000.000000ns

cap1                              =                          74
cap2                             =                         112
cap3                             =                         149
Duty                         =                    506.666656ns
Period                        =                    1000.000000ns

cap1                              =                          74
cap2                             =                         112
cap3                             =                         149
Duty                         =                    506.666656ns
Period                        =                    1000.000000ns
```

출력 결과

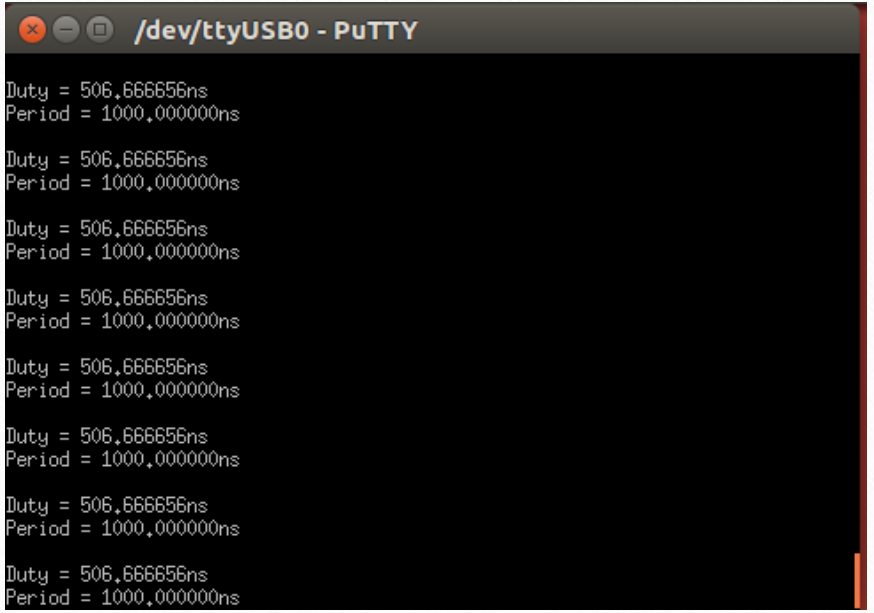
```
void ecapNotification(ecapBASE_t *ecap,uint16 flags)
{
    uint32 cap1, cap2, cap3;
    float64 duty, period;
    uint8 msg[128] = {0};

    cap1 = ecapGetCAP1(ecapREG1);
    cap2 = ecapGetCAP2(ecapREG1);
    cap3 = ecapGetCAP3(ecapREG1);

    duty = (cap2 - cap1)*1000/VCLK3_FREQ;
    period = (cap3 - cap1)*1000/VCLK3_FREQ;

    sprintf(msg,"Duty = %fns\r\n",duty);
    sciSend(sciREG1,strlen(msg),msg);
    sprintf(msg,"Period = %fns\r\n\n", period);
    sciSend(sciREG1,strlen(msg),msg);

}
```



문제점 및 진행계획

문제점

1. eCAP 개념 이해에 대한 어려움.
-> 데이터시트 해석으로 해결.

2. Halcogen 설정의 어려움
-> 데이터시트와 예제코드 활용하여 해결

3. printf 함수의 딜레이 현상
-> SCI(UART) 를 사용하여 해결

진행계획

eCAP을 활용하여 모터 속도 측정 및 제어

감사합니다