TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

2018-08-01 (4회차 중간발표)

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 - 정유경
ucong@naver.com

목차

ESC Calibration (with HobbyKing® ™ X-Car Beast Series ESC 1:8 Scale 150A)

변속기 즉 ESC(Electric speed controller) Calibration이란?

무선 조정기와 실제 ESC가 받아들이는 PWM 입력사이에 차이가 있으면 최대의 퍼포먼스가 나오지 않는다. 그래서 무선조정기와 ESC PWM 입력사이에 PWM 최소치와 최대치를 일치시키는 과정을 말한다. 변속기가 받을 신호의 최대점과 최저점을 기억시키고 구동 시 안정된 출력을 내게 하기 위한 것이다. 전원을 인가했을 때 모터가 갑자기 구동하지 않도록 Throttle 범위를 입력 받는 것이라고 생각해 볼 수 있다..

정확하게는 ESC throttle range calibration이라고 하며,
방법은 변속기 매뉴얼에서 찾아볼 수 있다.

사용한 ESC는 다음과 같다.
https://hobbyking.com/en_us/hobbykingr-tm-x-car-beast-series-esc-1-8-scale-150a.html

해당 ESC 매뉴얼에서 Calibration 부분을 찾아보자,

Throttle Range Calibration

1. Turn on the transmitter, then connect ESC with the battery packs and set the direction of the throttle channel to REV; set the EPA/ATV value of the throttle channel to 100%.
2. Press and hold the "Set" button and switch on the ESC, release the button when the Blue LED turn solid. Pull the throttle trigger to full position, Red LED light will flashes and motor beeps once. when system confirms the position.
3. Push the throttle trigger to full Brake position, Blue LED light will flashes and motor beeps twice when system confirms the position.
4. Now trigger goes back to neutral position, both of the Red LED and Blue LED blink and motor beeps three times when system confirms the position.
5. Turn off the ESC power switch to save the settings.
6. Turn the ESC back on. You are ready to use the ESC now.

Transmitter 가 있는 상태에서의 설정방법을 설명하고 있다.

우리는Transmitter의 Throttle대신, Putty 창에서 원하는 입력 값을 주고, MCU(TMS570LC43)의 PWM을 제어함으로써 캘리브레이션을 완료할 수 있다.

HalCoGen 설정하기

1. Driver Enable: SCI1, etPWM
2. Pinmux 설정: etPWM1A
3. GCM에서 VCLK3 14분주
4. Enable etPWM1 module
5. etPWM의 Clock configuration: ActualTBCLK = 1Mhz
6. PWM configuration: 주기 20ms, duty 1%

CCS 작성 #1

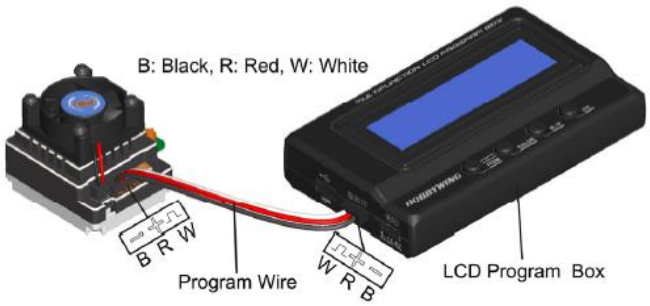
```
#include "HL_sys_common.h"
#include "HL_system.h"
#include "HL_etpwm.h"
#include "HL_sci.h"
#include <string.h>
#include <stdio.h>
#include <math.h>

#define TSIZE1 10
#define TSIZE2 5
#define TSIZE3 4
uint8 TEXT1[TSIZE1] = {'H', 'e', 'l', 'l', 'o', ' ', 'T', 'I', ' ', 'C', 'o', 'r', 't', 'e', 'x', '-', 'R', '5', 'F', ' ', 'M', 'C', 'U'};

unsigned int ePWM1B = 90;
void sciDisplayText(sciBASE_t *sci, uint32 *text, uint32 length);
void wait(uint32 time);

uint32 receiveData = 0;
uint32 temp = 0;
float flat = 1;
int ten = 10;
int squre = 3;

int main(void)
{
    sciInit(); /* SCI/SCI-Lin 초기화, 짝수 패리티, Stop Bits : 2 */
    etpwmInit();
```

```
    while(1)
    {
        etpwmStartTBCLK();
        squre = 3; // 10 의 제곱
        receiveData = 0;

        while(1)
        {
            temp = sciReceiveByte(sciREG1);
            sciDisplayText(sciREG1, &temp, TSIZE1); /* Text 전송 */
            temp = temp-48;
            if(temp > 9 || temp < 0)
                break;
            receiveData += temp * pow(ten, squre);
            printf("receiveData = %d₩n", receiveData);
            squre--;
        }

        if(receiveData > 0)
        {
            ePWM1B = receiveData;
            etpwmSetCmpB(etpwmREG1, ePWM1B);
            printf("ePWM1B = %d₩n", ePWM1B);
            wait(4000);
        }

        else        {
            printf("-------₩n");
        }
        wait(4000);
    }
}
/* scuREG1 , 48 or 49 or 50 , 10*/
void sciDisplayText(sciBASE_t *sci, uint32 *text, uint32 length){
    while(length--)
    {
        while((sciREG1->FLR & 0x4) == 4)
            ; /* wait until busy */
        sciSendByte(sciREG1, *text++); /* send out text */
```

```
    }
}

void wait(uint32 time){
    time--;
}
```

Throttle Range Calibration 방법은 다음과 같다.

1. 프로그램 카드를 연결하고 ESC와 배터리를 연결한다.
ESC를 켜고 "Ready to connect ESC" 확인
2. Throttle을 100%로 변경하고 Running Mode를 For/Rev로 설정한다.
(나머지는 Default 셋팅으로 진행)



3. 연결 해제하고 eTPWM1A(GIOA[5])와 ESC를 연결한다. → PUTTY를 켠다 → 0입력
(전원선(+)은 연결할 필요 없다)
2. Set 버튼을 홀드하고 ESC를 켠다. Blue LED가 안정적으로 켜지는 것이 확인되면 Set버튼에서 손을 뗀다.
3. PUTTY에 다음을 입력한다.
duty 2ms(Full Throttle) → Red LED flashes → Beep! X 1
duty 1ms(Full Brake) → Blue LED blinks → Beep! X 2
duty 1.5ms(Neutral) → both of the Red LED and Blue LED blink → Beep! X 3
적절한 타이밍이 중요하다. 0.5초 정도의 간격을 두고 셋팅한다.
6. ESC Power Off 하면 지금까지의 설정이 저장된다. <끝>

문제점

*. RTOS 올리면 RTI 사용이 불가능, 적절한 타이밍에 ESC에 MAX, MIN Value를 어떻게 입력할지 생각해보자
→ Delay를 주는 방법, Het을 사용하는 방법

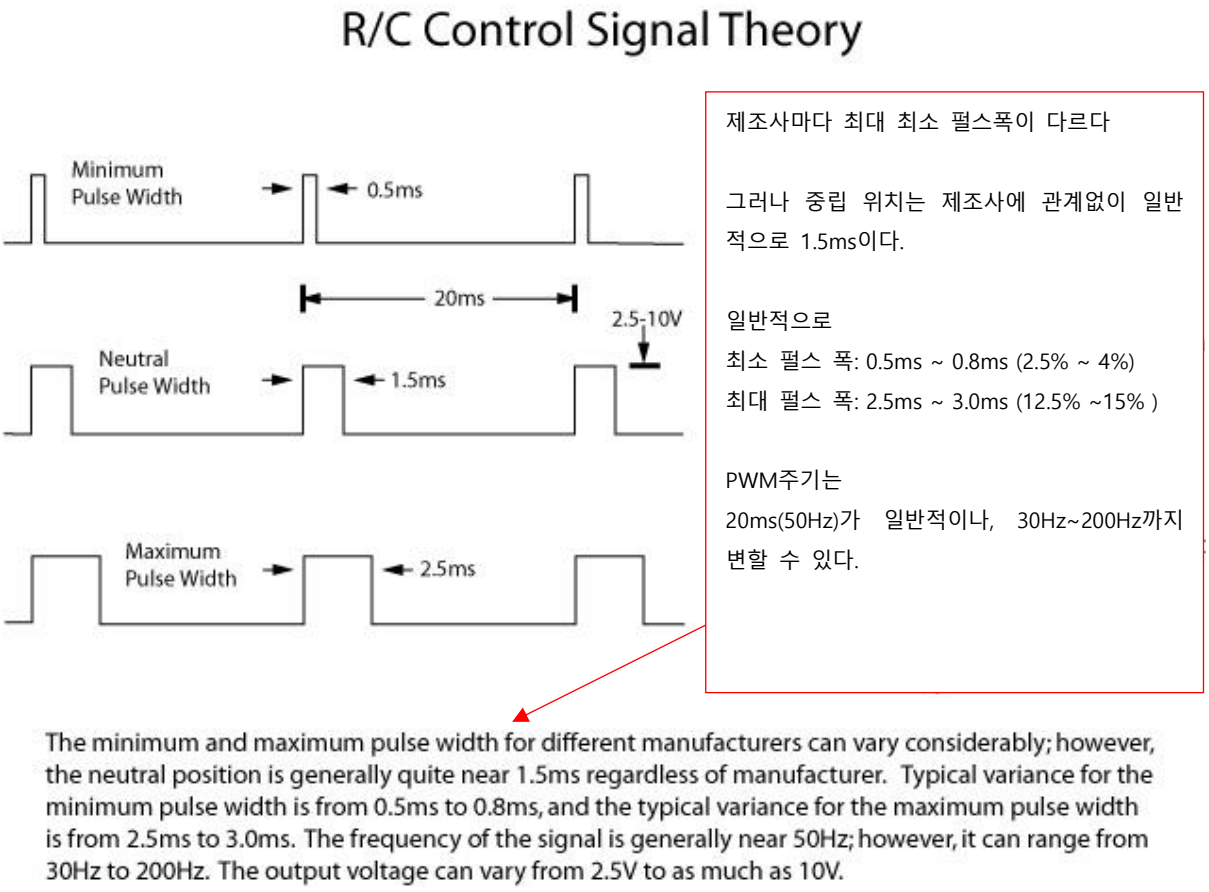
모터 구동 방법

1. PWM 2ms
2. PWM 1ms
3. 모터 작동을 위한 duty 입력

UART입력으로 PWM제어하면서 duty 조절해서 모터 구동범위 찾기

PWM 20ms 주기에 duty를 1ms, 2ms가 일반적인 RC Throttle의 범위이나, 제조사마다 Throttle의 동작범위가 다르기 때문에 다음과 같이 테스트를 진행해보았다.

다음의 코드를 이용하여
TimeBase Period = 25000 일때,
CompareA Value를 바꾸어 가면서 모터의 구동범위를 확인해 본다.



CCS 작성 #2

```
#include <HL_etpwm.h>
#include <HL_hal_stdtypes.h>
#include <HL_reg_sci.h>
#include <HL_sci.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

uint8 counter[4] = { 0 ,};
uint8 msg[32] = {0, };
void send_data(sciBASE_t *sci, uint8 *data,int length);
void main(void)
{
    int i;
    int real_value;
    uint8 value ;

    sciInit();
    etpwmInit();

    etpwmStartTBCLK();
/*          etpwmREG1->TBPRD = 24999U;
            etpwmREG1->CMPA = 1250U;       */
    while(1){
        for(i = 0 ; i < 4; i++){
            while(sciIsRxReady(sciREG1)==0)
                ;
             value = sciReceiveByte(sciREG1);
            // value -= 48 ; ASCII -> integer
             counter[i] = value;
             //sciSendByte(sciREG1, value);
             sprintf(msg,"input = %c   ",value);
             send_data(sciREG1, msg,strlen(msg));
        }
        sprintf(msg,"₩r₩n");
        send_data(sciREG1, msg,strlen(msg));

        sprintf(msg ,"real_value = %c%c%c%c₩r₩n",counter[0],counter[1],counter[2],counter[3]);
```

```
            send_data(sciREG1, msg,strlen(msg));

            real_value = atol(counter);

            if(real_value >0 && real_value <3750)
                etpwmREG1->CMPA = real_value ;
            else{
                sprintf(msg,"Fault₩r₩n");
                send_data(sciREG1, msg,strlen(msg));
            }
            //printf("%d₩n",real_value);
             //sciSendByte(sciREG1, uint8 byte);
        }
}

void send_data(sciBASE_t *sci, uint8 *data,int length)
{
    int i;
    for(i =0; i < length ; i++)
        sciSendByte(sci, data[i]);
}
```

이번주 목표

1. PWM Period를 5단계로 구동시킨다.
위의 실험을 통해서 적절한 값을 찾도록 한다.
2. RTI를 사용해서 ESC의 초기 Throttle설정을 자동으로 할 수 있도록 한다.
3. RTOS