< 8515 : PWM 신호 만들기 연구>

어셈은 안씁니다. C 만 가지고 Pocet-bot을 구현하는게 목표입니다. 물론, 하다가 벽에 부딪치게 되면 할 수 없이 어셈을 쓰게되는 불상사가 일어날 수도 있겠습니다만...

일단 어떤 기능을 작동시키는 시퀀스를 알기 위해서는 C 가지고 따라가 보는 것도 유용할 수 있겠죠?

컴파일러는 AVR-GCC입니다. 쉽게쉽게 할랍니다.

그리구... PWM(Pulse Width Modulation ; 펄스폭 변조) 에 관한 기본적인 설명은 생략합니다.

1. PWM 만들어 내는 표준적인 순서 파악하기

일단은, 8515에서 지원한다고 하는 2 채널의 PWM을 뽑아내기 위해서 어디어디를 건드려야 되는지를 파악해야겠네요. 데이터 시트가 어딨더라~ 요기 있네요...

네... 데이터 시트에 따르면, 8515는 타이머/카운터를 적절히 이용해서 PWM을 뽑아내도록 되어 있군요. 80196처럼 바로 C에서 불러다 쓸 수는 없네요... AVR-GCC 함수도 따로 준비되어 있는거 같지도 않고... 알아서 만들어 쓰라는 철저한 DIY 정신? DIE 되고 말겠당.

8비트 타이머/카운터 1개(0번 타이머/카운터), 16비트 타이머/카운터 1개(1번 타이머/카운터) 씩 제공되는데... 그중에 PWM 만들 때 쓰는 놈은 1번인 16비트 타이머/카운터네요.... 자.. 그럼 16비트 타이머/카운터가 설명되어 있는 페이지로 넘어가 봅니다. 엠... 몇 페이지랴? 34쪽이네요.

한번 대강 주욱 읽어 보니, 다음의 레지스터들을 건드리는게 셋팅하는 방법인가 봅니다. 그 근거는? 헤헷... 본문중에 'PWM' 이라는 글자가 중간중간 보이니까... 그게 맞겠죠. 쩝. 틀린가?

- TCCR1A / TCCR1B
- OCR1AH / OCR1AL

- OCR1BH / OCR1BL

뭐하는 것들인지는 영 잘 모르겠지만, 암튼 설명문 중에 'PWM' 이란 글자가 들어간 레지스터들입니다. 그리고... 40쪽에 대망의 'PWM 동작설명(Timer/Counter1 in PWM Mode)'가 나와 있군요. 제가 허접인 관계로, 이거 그냥 통째로 읽어 보겠습니다.

PWM 동작설명 (Timer/Counter1 in PWM Mode)

타이머/카운터 1번을 PWM 모드로 했을 경우를 볼짝시면, OCR1A (Output Compare Register 1A) 와 OCR1B (Output Compare Register 1B)... 요 두 놈을 이용해서 2개의 PWM 신호에 대해서 각각 8비트/9비트/10비트 분해능 선택이라든지, Free-running 이라든지, Glitch-free 라든지, Phase-correct 등을 맞출 수 있습니당. 이때 2개의 PWM 신호가 최종적으로다가 출력되는 핀은, PD5(OC1A) 및 OC1B 핀입니당. *8515 DIP 패키지에서는, 이것이 15번, 29번 핀이로구만요...*

요 모드에서 타이머/카운터 1번은 *업/다운* 카운터처럼 행동하게 되지요. 정확히 말하자면, $0000 에서부터 TOP (표 11 참조) 까지 업 되다가, 사이클이 반복되기 전에 다시 &0000 으로 향하게 다운 되어 갑니당. 그러니깐, 숫자가 삼각파 형태처럼 생겨나는 형상이로군요. 오르락~ 내리락~

카운터의 값이, OCR1A 및 OCR1B에 쓰여진 최고 10까지 설정되는 내용(분해능 설정)과 일치하게 되면... PWM 신호의 출력핀인 PD5(OC1A) / OC1B 들은, COM1A1 / COM1A0 및 COM1B1 / COM1B0 비트에 쓰여져 있는 내용을 참고해서 0 또는 1로 변화됩니다. 그게 한 사이클입니다. 아.... 이 비트들은, TCCR1A (Timer/Counter1 Control Register) 에 들어 있습니당. 표 12 보면서 이해해 보시죠. 끙끙... 이해하세요!

Table 11. Timer TOP Values and PWM Frequency

PWM Resolution	Timer TOP Value	Frequency
8-bit	$00FF (255)	$f_{TCK1}/510$
9-bit	$01FF (511)	$f_{TCK1}/1022$
10-bit	$03FF(1023)	$f_{TCK1}/2046$

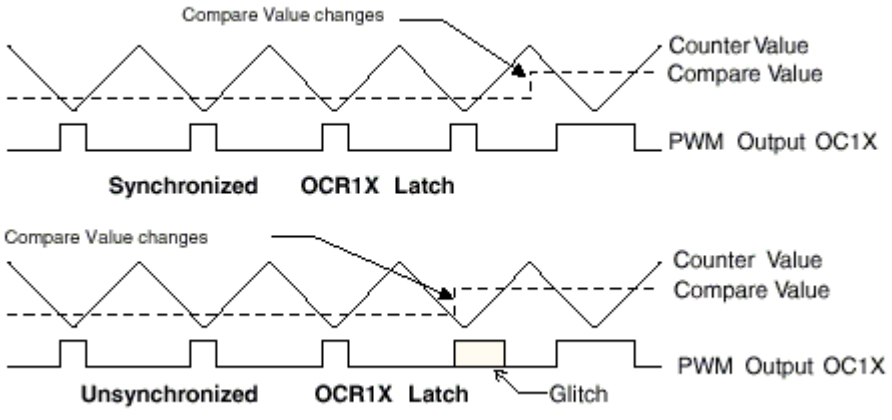
Table 12. Compare1 Mode Select in PWM Mode

COM1X1	COM1X0	Effect on OCX1
0	0	Not connected
0	1	Not connected
1	0	Cleared on compare match, up-counting. Set on compare match, down-counting (non-inverted PWM).
1	1	Cleared on compare match, down-counting. Set on compare match, up-counting (inverted PWM).

Note: X = A or B

PWM 모드에서, OCR1A / OCR1B 비트들에 내용을 쓰면 그것이 임시위치(Temporary Location)로 복사됩니다. 이것은 타이머/카운터 1번이 TOP 값에 다달았을 때, 래치됩니다. 이러한 동작을 함으로써, 동기가 되지 않았을때 OCR1A / OCR1B 에 내용을 써서.. 짝이 맞지 않는 PWM 파가 나오는 경우(요런걸 glitch 라고 하는군요)를 방지해 줍니다. 이해가 안되면 그림 32를 보시면서 머리를 짜내시길. 핫!핫!핫!

Figure 32. Effects on Unsynchronized OCR1 Latching



Note: X = A or B

뭐 별 건 아니네요... 그냥 펄스 하나가 튀어 올라오는거 막아준다는 소리네요.

다시 말하자면... 내용쓰기 하는 동작과 래치시켜주는 동작의 사이에는, OCR1A 및 OCR1B에서 읽는 것이 아니라 임시 위치 (Temporary location) 의 내용을 읽어서 사용한다는 소리죠... 이것은 즉, 가장 최근에 씌여진 내용은 대부분의 경우 OCR1A 및 OCR1B 바깥에 있고 바로 그것을 읽는다는 것입니다.

OCR1 의 내용이 $0000 이나 TOP 일 경우에는... 출력핀인 OC1A / OC1B 는 그 다음번 비교 동작에서 0또는 1로 업데이트 됩니당. 물론 이것은 COM1A1 / COM1A0 및 COM1B1 / COM1B0 비트들의 셋팅이 어떻게 되어 있느냐? 하는 것에 따라 결정되는 것이고요... 이 비트들의 설정은 아래 표 13을 보심 되겠습니당.

Note :

만약에 비교 레지스터가 TOP 값을 가지고 있고, 프리스케일러가 사용되지 않을 경우(CS12..CS10 = 001), PWM 출력은 나오지 않습니당. 뭐땀시 그러냐면, 업 카운팅 값과 다운 카운팅 값에 동시에 도달하게 되는 엽기적인 상황에 처하기 때문입니당. 프리스케일러가 사용되는 경우(CS12..CS10 ≠ 001 or 000)에는, 카운터가 TOP 값에 도달할 수 있기 땀시 PWM 이 동작할 수 있겠죠? 그러나! 다운 카운팅 비교는 도달되지 않기 때문에, 다음번 카운팅으로 TOP에 도달하기 전까지는 동작이 이어지지 않게쬬... 그래서 펄스가 딱 한 번 발생하고 맙니다. 이게 무슨 PWM 입니까.

Table 13. PWM Outputs OCR1X = $0000 or TOP

COM1X1	COM1X0	OCR1X	Output OC1X
1	0	$0000	L
1	0	TOP	H
1	1	$0000	H
1	1	TOP	L

Note: X = A or B

PWM 모드에서는, 카운터가 $0000 으로부터 카운팅 되기 시작할 때 TOV1 (Timer Overflow Flag 1) 이 1로 셋트됩니다. 글구, Timer Overflow Inturrupt 1 등은 보통의 타이머/카운터 모드에서와 같이 동작하게 되구요. 손볼 게 없다는 소리죠. 즉 TOV1 이 셋트되면서 Timer Overflow Inturrupt 1 가 깨어나 동작하게 된다는 소리.... 물론 이때 Timer Overflow Inturrupt 1 가 인터럽트 대기상태이고, 전체 인터럽트가 허용된 상태여야 함은 말할 나위도 없겠죠.

암튼 이런 글들을 읽어 보니 대략 감은 잡히네요. 삼각파를 만들어 비교하고, 이를 사용해 PWM을 만들어 낸다는 것은... 역시 정통적인 방법에 속하는 것인가 봅니다. 암튼 원리적인 거는 나중에 생각하기로 하고, 일단 Pocket-bot 이 필요로 하는 PWM을 만들기 위해서 필요한 초기화 과정을따져 봅니다.

1. TCCR1A 설정
2. TCCR1B 설정
3. OCR1AH / OCR1AL 설정
4. 인터럽트 인가

의 순서로 하면 무리가 없을 듯 합니다.

2. Pocket-bot에 적합하게 셋팅하기

Pocket-bot 은 2개의 모터가 달렸으니깐... PWM 2개 다 써야겠네용.

그리구, 분해능은... 그렇게 놓은 분해능까진 필요치 않을테니, 제일 낮은 8비트로 하구요.

이런 것들은... TCCR1A 레지스터에 관한 설명을 보믄서 맞춰 주면 끝이군요.

```
outp ( 0b10100001, TCCR1A );
```

이렇게요... 7,6번 및 5,4번 비트를 각각 0b10으로 써서 각 PWM 출력핀을 0으로 클리어 해줍니다. 그리구... 1,0번 비트는 0b01로 써서 PWM 모드로 결정지음과 동시에 8비트 분해능으로 설정.

이렇게 했으면, 이제 카운터/타이머 1번은 PWM 모드로 설정된 것으로 생각됩니다.

그리고, 이거가지고는 타이머/카운터 1번에 대한 설정을 다 못하니까, 8비트 레지스터를 더 준비해서 설정할 수 있도록 해 주고 있군요...

```
outp ( 0b00000001, TCCR1B );
```

상위 비트들은, 전부 일반 목적의 타이머/카운터로 쓸 때 사용할 것들이니까 0으로 맞추고... 하위 3비트 즉 2,1,0번 비트도 프리스케일링 설정하는 건데... 뭐 클럭을 분주할 일 없을테니 그냥 0b001로 맞추어 줌으로써 원래의 클럭을 그냥 사용하도록 합시다..

마지막으로, OCR1AH / OCR1AL 및 OCR1BH / OCR1BL 의 16비트 자릿수로 비교될 숫자를 설정해 주도록 되어 있군요... 뭐 앞에서 결정한 대로 8비트만 사용할 꺼니까, 상위의 OCR1AH 및 OCR1BH 는 전부 0으로 하고, 하위의 OCR1AL 및 OCR1BL 만 설정해 주면 되겠군요. PWM 값 변경할 경우에도 이놈만 변경해 주면 되는군요. 프로그램 처음에는 초기치 설정해 주는게 예의(?)니까,

```
outp ( 0b00000000, OCR1AH );   outp ( 0b11111111, OCR1AL );

outp ( 0b00000000, OCR1BH );   outp ( 0b11111111, OCR1BL );
```

이게 다인 것 같네요... (이상 적어본 Pseudo Code (가짜 프로그램 코드)상에서 2진수를 직접 적은 것은 단지 이해를 편하게 하기 위해서입니당~)

이상의 이해를 바탕으로... 간단히 PWM 출력해주는 함수를 적어봐야겠군요.

```
/* 쌍발 PWM 초기화하기~ */
int init_PWM(void)
{
    outp(0xA1,TCCR1A);    /* 분해능 8비트짜리 PWM 2개 준비하기 */
    outp(0x01,TCCR1B);    /* 프리스케일링 안하기 */
    outp(0x00,OCR1AH);    outp(OCR1AL,0xFF);    /* A 핀을 위한 초기값 적어주기 */
    outp(0x00,OCR1BH);    outp(OCR1BL,0xFF);    /* B 핀을 위한 초기값 적어주기 */
    sei();                          /* 인터럽트 인가하기 */
    return 0;
}

/* 쌍발 PWM 출력하기~ */
int out_PWM(unsigned char motor_a, unsigned char motor_b)
{
    outp(motor_a,OCR1AL);
    outp(motor_b,OCR1BL);
    return 0;
}
```

먼저 int_PWM() 함수를 호출하면 PWM 이 발동되는 거고... out_PWM() 함수로 원하는 값의 파를 마음껏 출력할 수 있겠군요...

이상의 방법은 너무 기본적인 거고... 만약에 PWM 이 더 많이 필요하게 되거나 좀 특이하게 해 보고 싶으면... 타이머/카운터를 잘 이용해서 다른 일반 출력 핀으로 신호를 만들어 뽑아내든지 하면 될테구요.. 거기에 관해서는 언급을 생략합니다.

그리구... 신호가 제대로 나오는지 않나오는지 확인해봤냐? 그런 의문도... 헐헐... 오시로스코프가 있어야 찍어보든 말든 할꺼 아닙니까. 전문적인 땜쟁이도 아닌데요. 암튼 나중에 직접 하드웨어 다 구성하고 나서, 동작을 보면서 확인해 볼 수밖에 없을꺼 같네요.