

NTC 8051 강좌 5. 타이머 인터럽트 사용하기

뉴테크놀로지 컴패니(N.T.C)

http://www.NTC.co.kr

1 서론

본 강좌에서는 타이머 인터럽트를 사용하는 방법에 관하여 설명하겠습니다. 타이머와 인터럽트를 설정하는 방법과 인터럽트 서비스 루틴 (ISR)을 프로그래밍 하는 방법에 관하여 설명하겠습니다. 타이머를 이용하여 1초에 1번씩 LED가 깜빡이는 것을 구현해 보겠습니다. 이를 응용하여 외부 인터럽트나 시리얼 인터럽트를 사용하시면 됩니다.

2 프로그래밍 방법

2.1 타이머 / 인터럽트 초기화

8051의 타이머는 비교적 간단한 구조를 가지고 있습니다. 타이머 또는 카운터로 사용할 수 있으며 8비트, 16비트 모드로 사용 가능합니다. 타이머의 동작 모드 설정은 TMOD 레지스터에서 할 수 있습니다.

TMOD 레지스터

7	6	5	4	3	2	1	0
GATE1	C/T1#	M11	M01	GATE0	C/T0#	M10	M00
Bit Number	Bit Mnemonic	Description					
7	GATE1	Timer 1 Gating Control Bit Clear to enable timer 1 whenever the TR1 bit is set. Set to enable timer 1 only while the INT1# pin is high and TR1 bit is set.					
6	C/T1#	Timer 1 Counter/Timer Select Bit Clear for timer operation: timer 1 counts the divided-down system clock. Set for Counter operation: timer 1 counts negative transitions on external pin T1.					
5	M11	Timer 1 Mode Select Bits M11 M01 Operating mode					
4	M01	0 0 Mode 0: 8-bit timer/counter (TH1) with 5-bit prescaler (TL1). 0 1 Mode 1: 16-bit timer/counter. 1 0 Mode 2: 8-bit auto-reload timer/counter (TL1). Reloaded from TH1 at overflow. 1 1 Mode 3: timer 1 halted. Retains count.					
3	GATE0	Timer 0 Gating Control Bit Clear to enable timer 0 whenever the TR0 bit is set. Set to enable timer/counter 0 only while the INT0# pin is high and the TR0 bit is set.					
2	C/T0#	Timer 0 Counter/Timer Select Bit Clear for timer operation: timer 0 counts the divided-down system clock. Set for counter operation: timer 0 counts negative transitions on external pin T0.					
1	M10	Timer 0 Mode Select Bit M10 M00 Operating mode					
0	M00	0 0 Mode 0: 8-bit timer/counter (TH0) with 5-bit prescaler (TL0). 0 1 Mode 1: 16-bit timer/counter. 1 0 Mode 2: 8-bit auto-reload timer/counter (TL0). Reloaded from TH0 at overflow. 1 1 Mode 3: TL0 is an 8-bit timer/counter. TH0 is an 8-bit timer using timer 1's TR0 and TF0 bits.					

TMOD 설정

TMOD 레지스터를 이용하여 Timer0 와 Timer1을 제어할 수 있습니다. 하위 4비트는 Timer0, 상위 4비트는 Timer1을 제어하게 됩니다. Timer1 은 시리얼 통신에서 Baud rate 을 만드는데 사용되기 때문에 여기서는 Timer0 를 사용하겠습니다.

여기서는 내부 클럭을 카운트 하는 타이머 모드로 동작을 하고 16비트 모드로 동작하게 됩니다.

```
TMOD = 0x01;           // 타이머 0 를 Mode1 16bit 모드로 사용
```

다른 부분에서 TMOD 를 설정할 경우 OR 연산을 해서 값을 설정해야 합니다.

Ex) TMOD |= 0x01;

타이머 간격 설정

다음으로 타이머가 얼마만큼 카운트 할지 설정합니다. 이것은 TH0 와 HL0 레지스터를 이용합니다. 8051의 타이머는 값을 증가시키면서 카운트를 하다가 0xffff 에서 0x0000 이 되는 순간 Flag 를 발생시킵니다. 따라서 0x1000 번 카운트 후에 동작을 하려면 0xffff - 0x1000 의 값을 TH0와 TL0에 넣어야 합니다.

여기서는 1ms 마다 동작하도록 설정하겠습니다.

X-Tal 속도 22.1184Mhz (AB-S52 개발보드 기본 장착)

실제 동작 속도 22.1184Mhz / 12 (내부에서 클럭을 12분주 함)

따라서 실제 동작 속도는 1.8432Mhz 로 이를 1843(0x733)번 카운트를 하면 1ms 가 됩니다. 따라서 0xffff - 0x733 의 값을 TH0와 TL0에 넣습니다.

```
TH0 = 0xff - 0x07;
```

```
TL0 = 0xff - 0x33;
```

타이머 동작 시작 시키기

타이머 동작을 시작하려면 TR0 레지스터를 1로 만들어야 합니다. TR0 는 TCON 레지스터의 5번째 비트로 비트단위의 제어가 가능합니다.

```
TR0 = 1;
```

인터럽트 활성화

타이머의 동작과 별도로 인터럽트의 동작을 활성화 해야 합니다. 8051은 5개의 인터럽트 소스를 가지며 이는 IE 레지스터와 IP 레지스터를 이용하여 제어합니다. IE 레지스터는 인터럽트 인에이블 레지스터로 각 인터럽트 소스별로 활성화 시킬 수 있습니다. 하나의 인터럽트라도 사용할 경우 최상위 비트는 1이 되어야 합니다. IP 레지스터는 인터럽트의 우선순위를 결정하는 레지스터로 여기서는 인터럽트를 하나만 사용하기 때문에 값을 지정하지 않아도 됩니다.

```
IE    = 0x82;
```

타이머 0 초기화 함수

```
void initTimer(void)
{
    TMOD |= 0x01;           // 타이머 0 를 Mode1 16bit 모드로 사용
    TH0  = 0xff-0x07;       // 1ms 설정
    TL0  = 0xff-0x33;
    TR0  = 1;
    IE   = 0x82;           // Timer0 Interrupt Enable
}
```

2.2 인터럽트 서비스 루틴(ISR) 선언하기

인터럽트 서비스 루틴을 선언하는 방법은 컴파일러마다 다릅니다. 이 부분은 표준화 되어 있지 않기 때문에 각 컴파일러에 맞게 매뉴얼을 참고하여 작성하면 됩니다. 선언을 한 후 동작하는 방법은 같습니다.

KEIL 컴파일러

KEIL 컴파일러에서는 아래와 같은 방법으로 선언을 할 수 있습니다. 함수를 구현한 후 함수의 선언 뒤쪽에 “interrupt” 키워드와 몇 번째 인터럽트인지 명시하면 됩니다.

```
void timer0_isr (void) interrupt 1
```

IAR 컴파일러

IAR 컴파일러에서는 아래와 같은 방법으로 선언을 할 수 있습니다. 함수를 구현한 후 함수의 선언 앞줄에 “#pragma” 와 함수의 선언 앞에 “__interrupt” 키워드를 명시하면 됩니다. “#pragma” 에서 “vector” 를 지정하는데 이는 어떤 인터럽트인지 명시하는 것입니다. “vector”의 값은 인터럽트 벡터 주소입니다. 타이머 0의 인터럽트 벡터는 0x0b 이므로 아래와 같이 쓰면 됩니다.

```
#pragma vector=0x0b
__interrupt void timer0_isr (void)
```

SDCC 컴파일러

SDCC 컴파일러에서는 아래와 같은 방법으로 선언을 할 수 있습니다. 함수를 구현한 후 함수의 선언 뒤쪽에 “__interrupt” 키워드와 몇 번째 인터럽트인지 명시하면 됩니다.

```
void timer0_isr (void) __interrupt (1)
```

2.3 인터럽트 서비스 루틴(ISR) 구현하기

타이머 값 재설정하기

타이머 인터럽트가 발생하는 것은 0xffff 에서 0x0000 으로 바뀌는 순간이므로 타이머의 값은 0x0000입니다. 따라서 원하는 시간 후에 다시 인터럽트를 발생시키려면 타이머 값을 재설정 해야 합니다.

```
TH0    = 0xff - 0x07;
TL0    = 0xff - 0x33;
```

LED 깜빡이기

타이머 인터럽트는 1ms 마다 발생하도록 되어 있습니다. 따라서 1초에 1번씩 깜빡이려면 0.5초간 켜졌다가 0.5초간 꺼지는 동작을 반복해야 합니다. 따라서 인터럽트가 500번 발생할 때마다 LED를 점멸하도록 하였습니다. 여기서 count 값은 전역 변수로 선언해야 다음 인터럽트가 발생하였을 때 값을 유지할 수 있습니다.

```
count ++;
if(count > 499){
    P0    ^= 0x03;           // 0, 1번 비트를 반전시킨다.
    count = 0;
}
```

인터럽트 서비스 루틴 (SDCC 컴파일러)

```
void timer0_isr (void) __interrupt (1)
{
    TH0    = 0xff - 0x07;
    TL0    = 0xff - 0x33;
    count ++;
    if(count > 499){
        P0    ^= 0x03;
        count = 0;
    }
}
```

3 주의 사항

1. 앞의 예제에서 “count” 값과 같이 인터럽트 서비스 루틴에서 값을 유지해야 하는 변수는 전역 변수를 사용해야 합니다.
2. 인터럽트 서비스 루틴과 메인 함수에서 모두 값을 쓸 경우 주의를 해야 합니다. 값을 쓰거나 읽는 중에 인터럽트가 발생하여 엉뚱한 값이 들어갈 수 있습니다. 실제 예러의 많은 부분이 여기서 발생합니다.
3. 인터럽트 서비스 루틴 안에서는 오래 걸리는 동작이나 무한루프, `printf` 와 같은 함수를 사용하지 않아야 합니다.