Chapter. 5

인터럽트

HBE-MCU-Multi AVR

국립 한밭대학교 HANBAT NATIONAL UNIVERSITY

Jaeheung, Lee

목차

1. 폴링과 인터럽트 그리고 인터럽트 서비스루틴
2. ATMega128 인터럽트
3. 인터럽트로 LED 점멸시키기
4. 인터럽트로 스톱워치 만들기

인터럽트

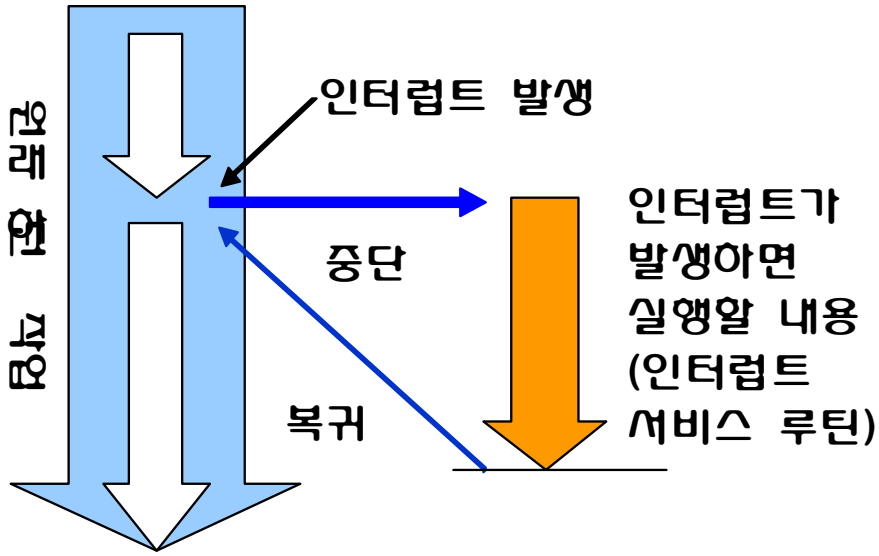
1. 폴링과 인터럽트 그리고 인터럽트 서비스루틴
2. ATMega128 인터럽트
3. 인터럽트로 LED 점멸시키기
4. 인터럽트로 스톱워치 만들기

인터럽트

- 인터럽트(Interrupt)
  - “방해하다” “훼방놓다”
  - 어떤 작업을 진행하고 있다가 갑자기 다른 일이 발생하여 먼저 처리해야 하는 상황을 **인터럽트 발생**이라 함.
  - 현재 수행중인 일을 잠시 중단하고 급한 일을 처리한 후 원래의 일을 다시 이어서 수행
  - 이때, 그 급한 일을 해결하는 것이 **인터럽트 서비스 루틴**임
  - 발생 시기를 예측할 수 없는 경우에 더 효율적

인터럽트

- 인터럽트와 인터럽트 서비스 루틴
  - 인터럽트가 발생하면 프로세서는 현재 수행중인 프로그램을 멈추고
  - 상태 레지스터와 PC(Program Counter)등을 스택에 잠시 저장한 후 인터럽트 서비스 루틴으로 점프한다.
  - 인터럽트 서비스 루틴을 실행한 후에는 이전의 프로그램으로 복귀하여 정상적인 절차를 실행한다.



폴링과 인터럽트

- 마이크로 컨트롤러의 외부 상황 입력 방법
  - **폴링(polling)** : 사용자가 명령어를 사용하여 입력 핀의 값을 계속 읽어서 변화를 알아내는 방식
  - **인터럽트(interrupt)** : MCU 자체가 하드웨어적으로 그 변화를 체크하여 변화시에만 일정한 동작을 하는 방식

인터럽트

인터럽트의 구성요소

- **발생원** : 누가 인터럽트를 요청했는가?
- **우선순위** : 2개 이상의 요청시 누구를 먼저 서비스 할까?
  (중요도 : Priority)
- **인터럽트벡터** : 서비스루틴의 시작번지는 어디인가?

인터럽트

- 인터럽트의 종류
  - 발생원인에 따른 인터럽트 분류
    - 내부 인터럽트
    - 외부 인터럽트
  - 차단가능성에 의한 인터럽트 분류
    - 차단(마스크) 불가능(Non maskable, NMI)인터럽트
    - 차단(마스크) 가능(Maskable)인터럽트
  - 인터럽트 조사 방식에 따른 분류
    - 조사형 인터럽트(Polled Interrrupt)
    - 벡터형 인터럽트(Vectored Interrupt)

ATMega128 인터럽트

- ATMega128 인터럽트
  - 차단 가능한 외부 인터럽트
  - 리셋 포함 총 35개의 인터럽트 벡터를 가짐
    - 리셋 1개
    - 외부핀을 통한 외부 인터럽트 8개
    - 타이머 관련 14개
      - 타이머 0(2개), 타이머 1(5개), 타이머 2(2개), 타이머 3(5개),
    - UART 관련 6개
      - USART0(3개), USART1(3개),
    - 기타 6개

ATMega128 인터럽트

- ATMega128 인터럽트
  - 모든 인터럽트는 전역 인터럽트 인에이블 비트인 SREG의 I 비트와 각각의 개별적인 인터럽트 플래그 비트가 할당되어 있다.
  - 인터럽트들과 개개의 리셋벡터는 각각 개별적인 프로그램 벡터를 프로그램 메모리 공간내에 가진다.
  - 모든 인터럽트들은 개별적인 인터럽트 허용 비트를 할당 받는다.
  - 특정 인터럽트를 가능하게 하려면 특정 인터럽트 가능 비트와 상태레지스터(SREG)에 있는 전체 인터럽트 허용 비트(I 비트)가 모두 1로 세트되어 있어야 한다.

ATMega128 인터럽트

- 상태레지스터(SREG:Status REGister)
  - ALU의 연산 후 상태와 결과를 표시하는 레지스터

7	6	5	4	3	2	1	0
I	T	H	S	V	N	Z	C

비트	설명
I	Global Interrupt Enable
T	Bit Copy Storage
H	Half Carry Flag
S	Sign Bit
V	2's Complement Overflow Flag
N	Negative Flag
Z	Zero Flag
C	Carry Flag

ATMega128 인터럽트

- 인터럽트 마스크 레지스터
  (EIMSK : External Interrupt MaSK register)
  - 외부 인터럽트의 개별적인 허용 제어 레지스터
  - INTn이 1로 세트되면 외부 인터럽트 인에이블

7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

비트	설명
INT7~0	Int7~0 Interrupt Enable

ATMega128 인터럽트

ATMega128 인터럽트 우선순위

- 프로그램 메모리 공간에서 최하위 주소는 리셋과 인터럽트 벡터로 정의되어 있다.
- 리스트는 서로 다른 인터럽트들의 우선순위를 결정한다.
- 최하위 주소에 있는 벡터는 최상위 주소에 있는 벡터에 비해 우선순위가 높다.
  - RESET : 최우선 순위
  - INT0(External Interrupt Request 0) : 2순위
- MCU 제어 레지스터(MCUCR: MCU Control Register)의 IVSEL(Interrupt Vector SELect) 비트를 세팅함으로써 인터럽트 벡터의 배치를 변경할 수 있다.

ATMega128 인터럽트

외부 인터럽트의 트리거

- 인터럽트를 발생시키기 위한 "방아쇠"
- 인터럽트 발생의 유무를 판단하는 근거가 됨.
- 트리거 방법
  - Edge Trigger : 입력 신호가 변경되는 순간을 인터럽트 트리거로 사용하는 경우
    - 하강에지(Falling Edge) 트리거 : '1'에서 '0'로 변경되는 시점을 사용
    - 상승에지(Rising Edge) 트리거 : '0'에서 '1'로 변경되는 시점을 사용
    - ATMega128에서 에지 트리거는 50ns 이상의 펄스폭을 가져야 한다.
  - Level Trigger : 입력 신호가 일정 시간동안 원하는 레벨을 유지되면 트리거 하는 경우
    - 평상시 High(1)로 있다가 Low(0)로 변화되어 일정시간 유지되면 트리거 하게 됨.
    - 레벨 트리거 인터럽트 신호는 워치독 오실레이터에 의해 2번 샘플링되며 이 기간이상의 펄스폭을 주어야한다.

ATMega128 인터럽트

- EICRA(External Interrupt Control Register A)
  - 외부 인터럽트 0~3의 트리거 설정에 사용

7	6	5	4	3	2	1	0
ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00

ISCn1	ISCn0	설명
0	0	INT의 Low level에서 인터럽트를 발생한다.
0	1	예약
1	0	INT의 하강 에지에서 인터럽트를 발생한다.
1	1	INT의 상승 에지에서 인터럽트를 발생한다.

ATMega128 인터럽트 트리거

- EICRB(External Interrupt Control Register B)
  - 외부 인터럽트 4~7의 트리거 설정에 사용

7	6	5	4	3	2	1	0
ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40

ISCn1	ISCn0	설명
0	0	INTn1의 Low level에서 인터럽트를 발생한다.
0	1	INTn 핀에 논리적인 변화가 발생할 경우
1	0	INT의 하강 에지에서 인터럽트를 발생한다.
1	1	INT의 상승 에지에서 인터럽트를 발생한다.

ATMega128 인터럽트 트리거

- EICRB(External Interrupt Control Register B)
  - 외부 인터럽트 4~7의 트리거 설정에 사용

7	6	5	4	3	2	1	0
ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40

ISCn1	ISCn0	설명
0	0	INTn1의 Low level에서 인터럽트를 발생한다.
0	1	INTn 핀에 논리적인 변화가 발생할 경우
1	0	INT의 하강 에지에서 인터럽트를 발생한다.
1	1	INT의 상승 에지에서 인터럽트를 발생한다.

ATMega128 인터럽트 동작

- EIFR(Interrupt Flag Register)
  - 외부 인터럽트 발생 여부를 알려주는 레지스터
  - 외부 인터럽트가 에지 트리거에 의해 요청된 경우 허용여부에 상관없이 1로 세트

7	6	5	4	3	2	1	0
INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	INTF0

비트	설명
INTF7~0	IntF7~0 Interrupt Flag

ATMega128 인터럽트

ATMega128 인터럽트 프로그램

인터럽트 프로그램의 틀

```
#include <io.h>
#include <interrupt.h>        /* 인터럽트 관련 시스템 헤더 파일  */
SIGNAL(SIG_INTERRUPT0)   /* Int0에 대한 Interrupt Service Routine 선언*/
{
      /* 인터럽트가 발생되었을 때 실행할 명령어 세트를 선언 */
}
int main(void)
{
DDRD = 0xFE;                    /* 인터럽트 입력으로 D 포트를 사용*/
EIMSK = 0x01;                   /* external interrupt 0 을 Enable*/
sei();                        /* 전체 인터럽트 Enable */
for (;;) {                      /*  main loop 명령어 세트 */
 }
```

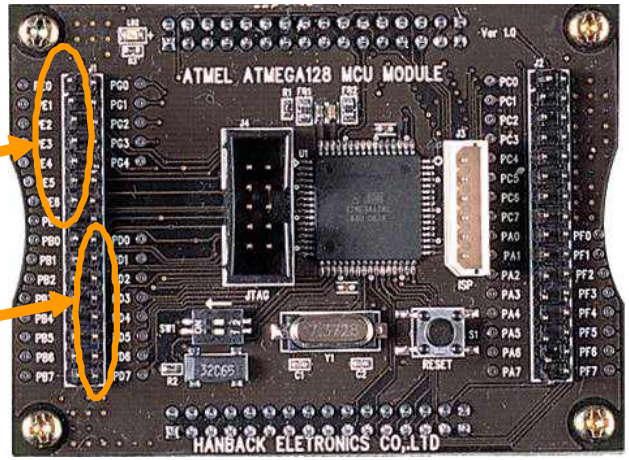
실습 5 : 인터럽트로 LED 점멸

- 실습 개요
  - ATmega128 마이크로컨트롤러의 인터럽트 기능을 이용하여 LED를 점멸시키는 실습
  - 일정시간 마다 LED가 순차적으로 켜지도록 하고, 버튼 스위치를 누르면 LED가 멈추었다가 다시 누르면 동작하도록 한다.
  - 버튼 스위치가 눌려지면 인터럽트가 발생하도록 해야 한다.
  - 입력포트 1개(인터럽트가 가능한 포트), 출력포트 1개 사용

- 실습 목표
  - 인터럽트 발생 원리 이해
  - 인터럽트 제어 방법의 습득(관련 레지스터 이해)
  - 입출력 포트에 관한 이해(특히 인터럽트 관련 포트)

실습 5 : 인터럽트로 LED 점멸

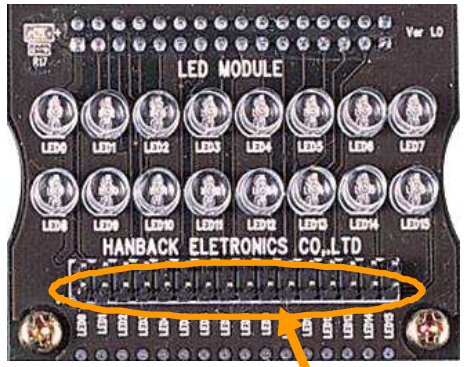
□ 사용 모듈 : MCU 모듈, LED 모듈, Switch 모듈



MCU 모듈 포트 E

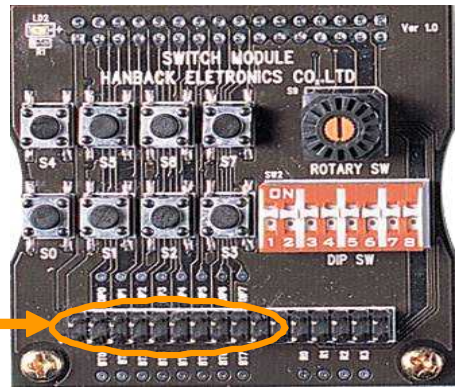
MCU 모듈 포트 D

MCU 모듈



LED 모듈

LED 모듈 Signal

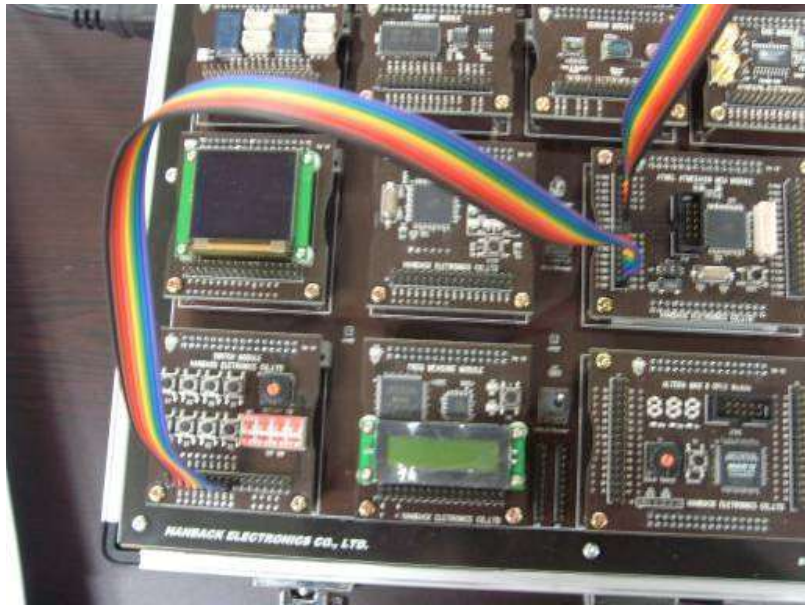


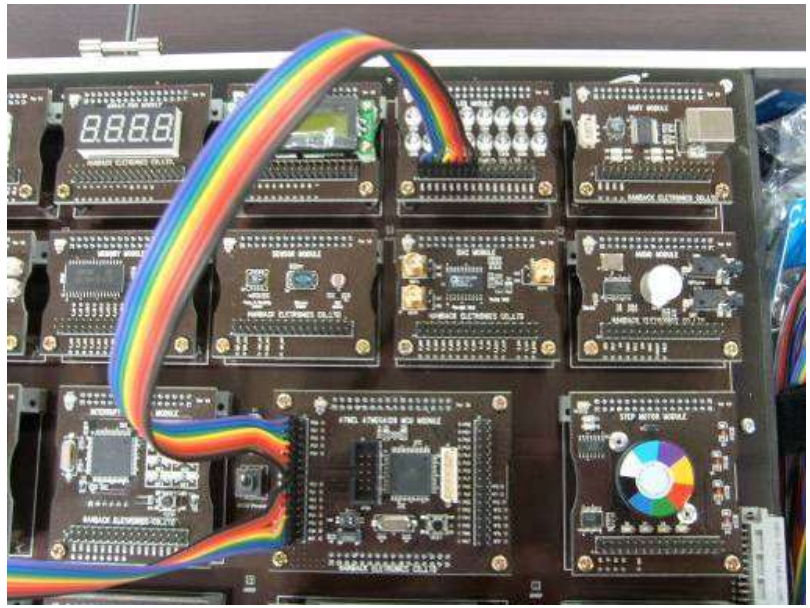
Switch 모듈 버튼 스위치 Signal

Switch 모듈

실습 5 : 인터럽트로 LED 점멸

- 모듈 결선 방법
  - MCU 모듈 포트 E의 PE0 ~PE7을 LED 모듈의 LED 0 ~ 7까지 연결
  - MCU 모듈 포트 D의 PD0를 2핀 케이블로 Switch 모듈의 BT0에 연결





실습 5 : 인터럽트로 LED 점멸

- 구동 프로그램 : 사전 지식
  - 인터럽트를 위한 입력 포트 선택
    - 포트 D의 0번 비트는 Int0로서, 인터럽트로 사용할 수 있는 포트임.
  - 외부 인터럽트 Enable
    - 상태 레지스터(SREG)의 전체 인터럽트 허용비트(I 비트) 를 '1' 로 세팅
    - Int0의 인터럽트 마스크 레지스터의 0번 비트를 1로 세팅.
  - 인터럽트 트리거 방법 정의
    - 상승 에지에서 트리거하도록 EICRA 레지스터를 세팅.
  - 전체 인터럽트 Enable
    - sei();
  - Main 루틴 기술
  - 인터럽트 서비스 루틴의 선언
    - SIGNAL(인터럽트소스명); 방식을 사용

실습 5 : 인터럽트로 LED 점멸

- 구동 프로그램 : 소스 분석
  - Interrupt_led.c

1)	#include<avr/io.h> #include<avr/interrupt.h> #include<util/delay.h> unsigned char Time_STOP = 0;

2)	SIGNAL(SIG_INTERRUPT0); /* 인터럽트 서비스 루틴 선언 */
	int main(){
	unsigned char LED_Data = 0x01;

3)	DDRD = 0xFE; /* 포트 D의 0 번째 레지스터를 사용하여 입력 (0xFE는 1~7비트까지의 레지스터를 의미) */ DDRE = 0xFF; /* 포트 E의 0~7번째까지의 모든 레지스터를 출력으로 사용 */

실습 5 : 인터럽트로 LED 점멸

4)	EICRA = 0x0F; /* 0~3비트까지 "1" 로 두어 인터럽트0에서 상승 에지를 발생한다. (EICRA: Interrupt sense control 및 MCU의 일반적인 기능을 설정하는데 사용) */ EICRB = 0x00; EIMSK = 0x01; /* 0비트가 "1" 로 셋되고, SREG 레지스터의 I비트가 "1" 로 설정되어 있으면 외부인터럽트는 enable된다. (EIMSK: INT0~INT7의 개별 인터럽트를 설정) */ EIFR = 0x01; /* 0비트가 "1" 로 셋되고, SREG 레지스터의 I비트와 EIMSK 레지스터의 INT7~INT0비트가 "1" 로 설정되어 있으면, MCU는 해당하는 인터럽트 벡터로 점프한다. (EIFR: EIMSK 레지스터에서 설정한 개별 인터럽트의 상태를 나타냄) */
5)	sei(); while(1){

실습 5 : 인터럽트로 LED 점멸

6)

```
PORTE = LED_Data;     /* 포트E를 LED_Data로 두고, LED_Data를 하나씩 쉬프트
                         시킨다.  */
if(Time_STOP == 0) {
   if(LED_Data == 0x80)  LED_Data = 0x01;
   else LED_Data <<= 1;
 }
_delay_ms(100);
}
return 0;
}
```

7)

```
SIGNAL(SIG_INTERRUPT0){      // Stop/Resume 처리
cli();
if(Time_STOP == 0)
Time_STOP = 1;
else
Time_STOP = 0;
sei();
}
```

인터럽트 서비스 루틴

실습 5 : 인터럽트로 LED 점멸

- 실행 결과

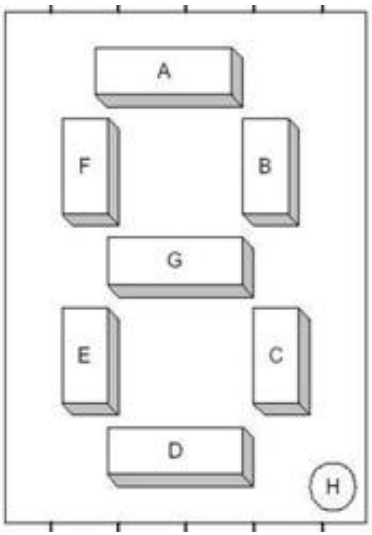
실습 6 : 인터럽트를 이용한 스톱워치

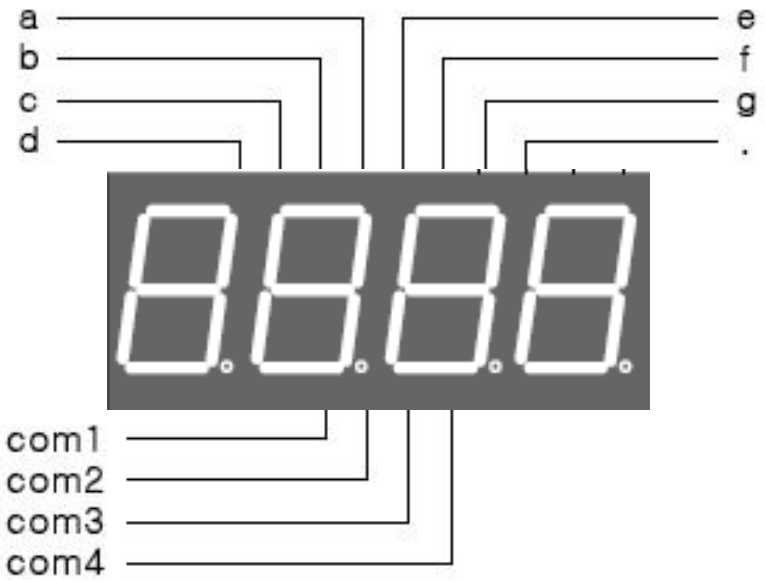
- 실습 개요
  - 스위치 모듈과 Array-FND 모듈에 연결하여 스톱 워치를 제작
  - 일정 시간마다 클럭에 의해 FND에 숫자와 문자가 디스플레이 되도록 하고, 스위치를 누르면 FND 디스플레이가 초기화되도록 하며, 또한 다른 버튼을 누르면 잠시 멈추었다가 다시 이어서 동작을 하도록 한다.
  - 포트 D의 0번 비트와 1번 비트를 Int0와 Int1의 인터럽트로 사용
    - Int0에 의해서 스톱워치의 Stop/Resume 기능을 구현.
    - Int1을 이용하여 스톱워치의 리셋 기능을 구현.
  - 스톱워치의 표시는 Array FND 를 사용한다.

- 실습 목표
  - 인터럽트 활용 방법의 습득(관련 레지스터 이해)
  - Array FND 동작 원리 이해

실습 6 : 인터럽트를 이용한 스톱워치

- 7-Segment FND Array
  - 스톱워치등 7-segment FND 를 여러 개 Array로 묶어서 사용
  - 7-Segment의 출력은 공통으로 연결하고, 출력할 위치를 지정하는 com1 ~ com4의 값을 제어하여 원하는 숫자나 문자를 표시

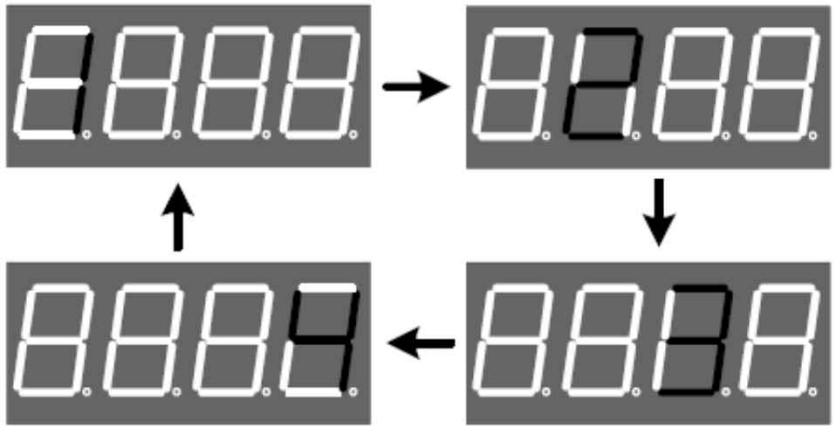




실습 6 : 인터럽트를 이용한 스톱워치

- 7-Segment FND Array 구동원리
  - 4개의 7-Segment에 "1234" 의 숫자를 표시하기 위한 방법

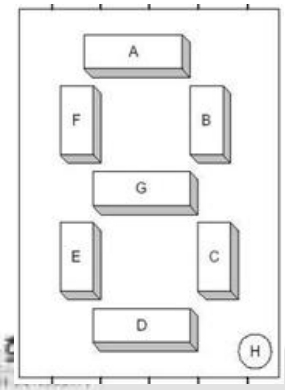
데이터 : 00000110
Com 1~4 : 0111



데이터 : 01011011
Com 1~4 : 1011

데이터 : 01100110
Com 1~4 : 1110

데이터 : 01001111
Com 1~4 : 1101

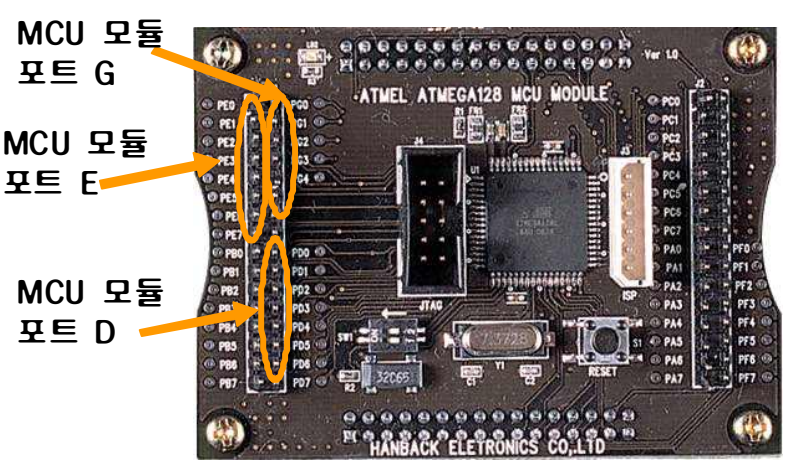


H	G	F	E	D	C	B	A
0	1	0	0	1	1	1	1

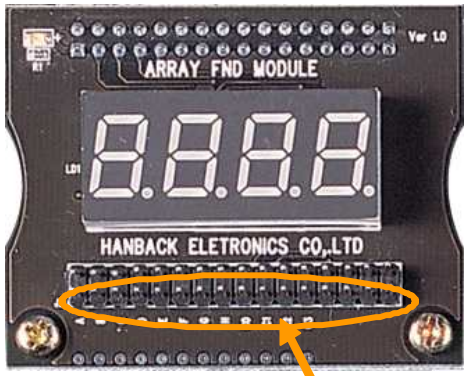
Com1	Com2	Com3	Com4
1	1	0	1

실습 6 : 인터럽트를 이용한 스위치

□ 사용 모듈 : MCU 모듈, ArrayFND 모듈, Switch 모듈

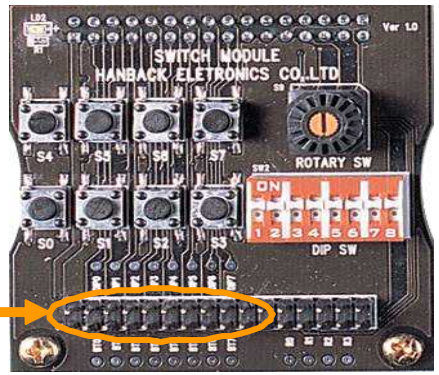


MCU 모듈



ArrayFND 모듈

ArrayFND
모듈 Signal

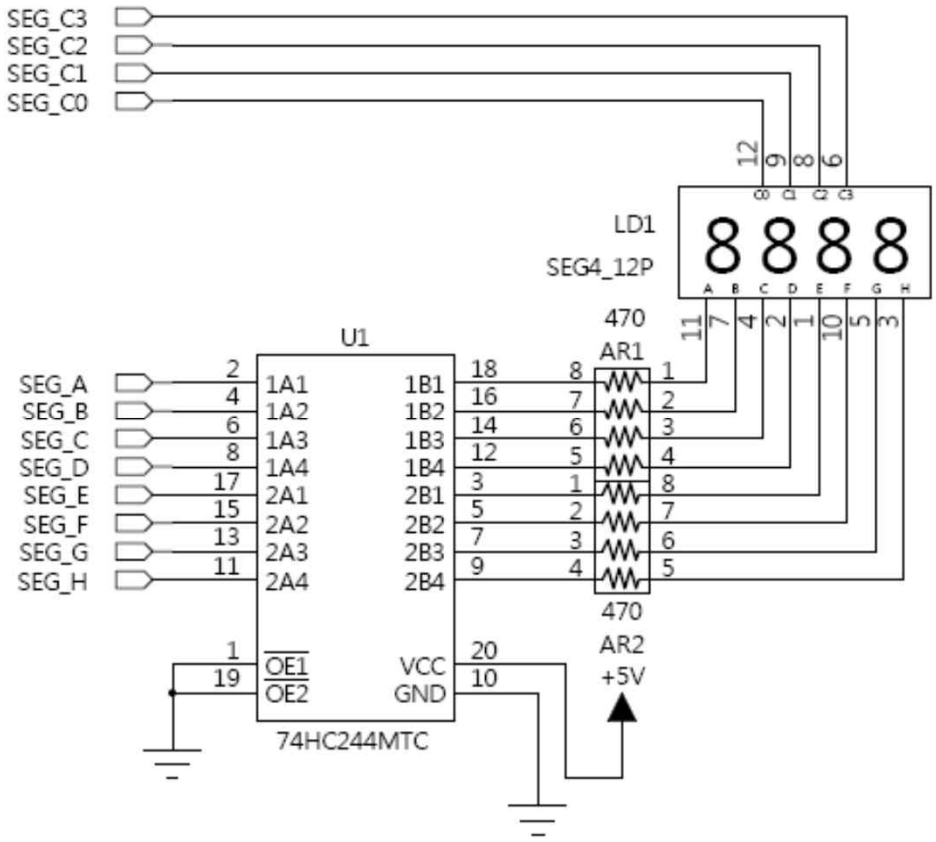


Switch 모듈
버튼 스위치 Signal

Switch 모듈

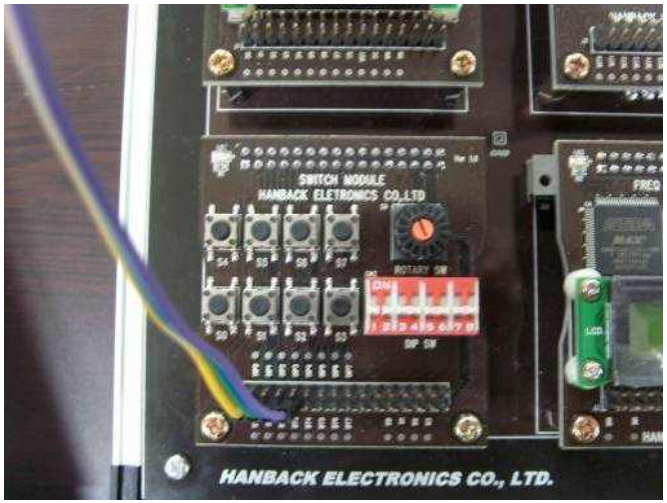
실습 6 : 인터럽트를 이용한 스톱워치

- Array-FND 모듈의 회로도

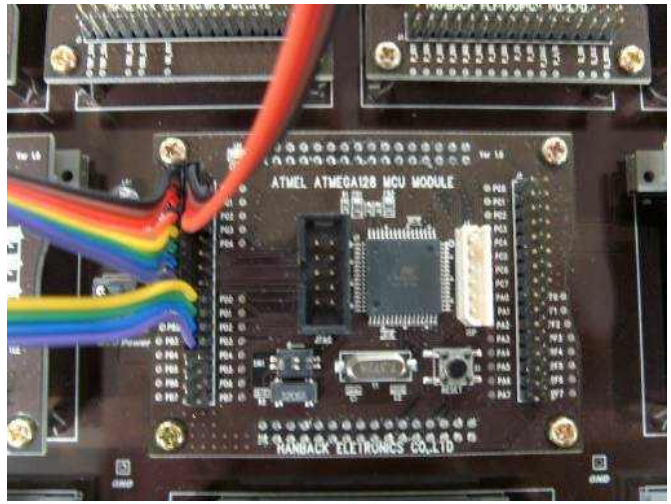


실습 6 : 인터럽트를 이용한 스톱워치

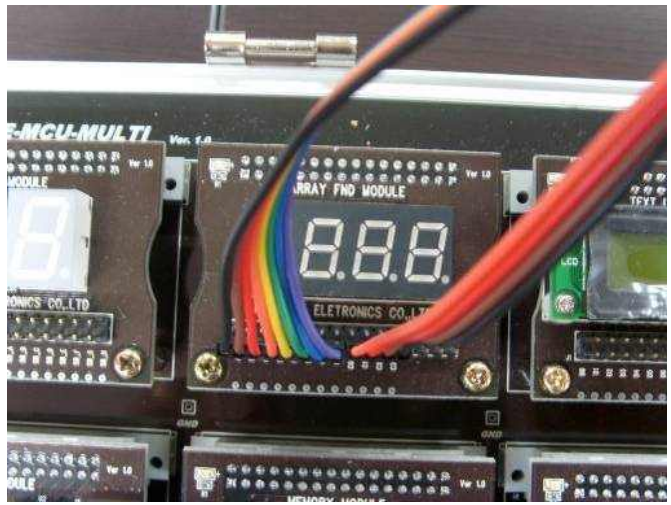
□ 모듈 결선 방법



MCU 모듈 포트 D의 PD0와 PD1을 Switch 모듈의 BT0 및 BT1에 연결



MCU 모듈 포트 G의 PG0~PG3는 Array FND 모듈의 C0~C3 포트에 연결



MCU 모듈 포트 E의 PE0~PE7를 Array FND 모듈의 A~H까지 연결.

실습 6 : 인터럽트를 이용한 스톱워치

- 구동 프로그램 : 사전 지식
  - 인터럽트를 이용하여 스톱 워치와 비슷한 기능을 하도록 만든 것이다
  - FND Array는 A~H까지의 각 7-Segment의 LED를 켜기 위한 신호를 사용
    - 위치선택 신호(C3~C0)
    - 잔상 효과를 위해 약간의 시간지연이 필요
  - C3~C0는 MCU G포트의 PG3~PG0에 연결되어 있고, A~H 포트는 MCU E 포트의 PE7~PE0 연결
  - 입출력 포트 D의 0번 비트는 Int0로서, 1번 비트는 Int1으로 사용
  - 스톱워치와 유사하게 기능을 하도록 만들어진 것으로 실제로 정확한 시간을 잴 수 있는 완전한 스톱워치가 아님
    - 메인 함수의 While-Loop를 한바퀴 돌아온 시간을 10ms으로 간주
    - 상대적인 시간

실습 6 : 인터럽트를 이용한 스톱워치

- 구동 프로그램 : 소스 분석
  - Interrupt_stopwatch.c

1)	#include<avr/io.h>
	#include<avr/interrupt.h>
	#include<util/delay.h>
	unsigned char time_10ms=0,time_100ms=0,time_1s=0,time_10s=0;
	char Time_STOP = 0;

2)	SIGNAL(SIG_INTERRUPT0); SIGNAL(SIG_INTERRUPT1); int main(){ unsigned char FND_DATA_TBL[] = {0x3F, 0X06, 0X5B, 0X4F, 0X66, 0X6D, 0X7C, 0X07, 0X7F, 0X67, 0X77, 0X7C, 0X39, 0X5E, 0X79, 0X71, 0X08, 0X80};

3)	DDRD = 0xFC; /* 포트 D의 0,1 번째 레지스터를 사용하여 입력(0xFC는 2~7 비트까지의 레지스터를 의미) */
	DDRG = 0x0F; // 포트 G의 0~3번째까지의 레지스터를 출력으로 사용
	DDRE = 0xFF; // 포트 E의 0~7번째까지의 모든 레지스터를 출력으로 사용

실습 6 : 인터럽트를 이용한 스톱워치

4)	EICRA = 0x0F; /* 0~3비트까지 "1" 로 두어 인터럽트0과 1에서 상승 에지를 발생한다. (EICRA: Interrupt sense control 및 MCU의 일반적인 기능을 설정하는데 사용) */ EICRB = 0x00; EIMSK = 0x03; /* 0~1비트까지 "1" 로 셋되고, SREG 레지스터의 I 비트가 "1" 로 설정되어 있으면 외부인터럽트는 enable된다. */ // (EIMSK: INT0~INT7의 개별 인터럽트를 설정) EIFR = 0x03; /* 0~1비트까지 "1" 로 셋되고, SREG 레지스터의 I비트와 EIMSK 레지스터의 INT7~INT0비트가 "1" 로 설정되어 있으면, MCU는 해당하는 인터럽트 벡터로 점프한다. /* (EIFR: EIMSK 레지스터에서 설정한 개별 인터럽트의 상태를 나타냄) */
5)	sei(); while(1){

실습 6 : 인터럽트를 이용한 스톱워치

6)	PORTG = 0x07; // C3을 선택한다
	PORTE = FND_DATA_TBL[time_10ms]; /* FND_DATA_TBL배열에서 time_10ms 만큼의 FND데이터를 출력한다. */
	_delay_ms(2); //잔상을 남게하기 위한 딜레이
	PORTG = 0x0B; //C2를 선택
	PORTE = FND_DATA_TBL[time_100ms];
	_delay_ms(2);
	PORTG = 0x0D; //C1을 선택
	PORTE = FND_DATA_TBL[time_1s]|0x80;
	_delay_ms(3);
	PORTG = 0x0E; //C0을 선택
	PORTE = FND_DATA_TBL[time_10s];
	_delay_ms(3);
	if(Time_STOP == 1) continue; // 인터럽트에 의한 Stop/Resume 처리
	/*여기까기 사용된 딜레이 함수가 약 10ms이므로 time_10ms는 10ms마다 증가된다. */

실습 6 : 인터럽트를 이용한 스톱워치

6)

```
time_10ms++;
if(time_10ms == 10){
time_10ms = 0;
time_100ms++ ;
}
if(time_100ms == 10){
time_100ms = 0;
time_1s++ ;
}
if(time_1s == 10){
time_1s = 0;
time_10s++ ;
}
if(time_10s == 10){
time_10s = 0;
}
}
return 0;
}
```

실습 6 : 인터럽트를 이용한 스톱워치

7)	SIGNAL(SIG_INTERRUPT0){ // Stop/Resume 처리
	cli();
	if(Time_STOP == 0)
	Time_STOP = 1;
	else
	Time_STOP = 0;
	sei();
	}

8)	SIGNAL(SIG_INTERRUPT1){ //리셋
	cli();
	time_10ms = 0;
	time_100ms = 0;
	time_1s = 0;
	time_10s = 0;
	sei();
	}

인터럽트 서비스 루틴 0 :
스톱워치의 Stop/Resume 처리

인터럽트 서비스 루틴 1 :
스톱워치의 Reset 처리

실습 6 : 인터럽트를 이용한 스톱워치

- 실행 결과