AVR - HW5

임베디드스쿨1기
Lv1과정
2020. 10. 17
박성환

1. Review-Struct/union

- **struct/union 내것 만들기**

- 보면 내용 이해는 충분히 하지만 안보고 당연히 써야지 생각될정도로 만들기

```
/* 구조 이해는 충분히 하지만 아애 암기해버리기*/
struct adc{
    union
    {
        struct
        {
            uint8_t adc_l;
            uint8_t adc_h;
        };

        uint16_t adc;
    };
    uint8_t adcsr_a;
    uint8_t adcsr_b;
    uint8_t admux;
};
```

1. Review-const

- **Const 위치에 따른 해석**

- 매번 햇갈린 부분이지만 이번에 한 번 정리해두기

```
* 후치 방식
* int const *p1; //const가 int 수식 => *p1 값 상수화, (다)안됨
* int * const p2; //const가 int* 수식 => 주소 p2 상수화, (가, 나)안됨
* int const * const p3; //const가 int 수식 & const가 포인터 수식 => *p3값, p3 주소 전부 상수화, (가,나,다)안됨
* const int * p4; //맨위의 방식의 const와 int 자리만 바뀐것 (자주씀, (다)안됨))

* 아래 가능한것 찾기
*  (가) p = (int*) 0x1000;
*  (나) p++;
*  (다) *p=100;
*  (라) a = *p;
*
volatile struct adc *const adc = (void*)ADC_REG; //volatile, const 필요성 상기
```

1. Review-itoa/sprintf

1) itoa() 함수

- stdlib.h를 선언한다. #include <stdlib.h>
- **사용법)** itoa(변환할 정수, 문자열로 변환한 값을 받을 문자열 변수, 진법 선택)
  itoa(int cnt, char * buffer,int n)
- ex)

```
int cnt=1;
char buffer[10];

itoa(cnt,buffer,10); // 숫자 1을 문자열로 변환하여 buffer 변수에 저장. 이 때 숫자는 10진수로 변환한다.
```

Itoa함수는 표준함수가 아니라 win32api라서 리눅스에서는 사용 불가능

참고로 atoa는 표준함수로 어디든 사용 가능

2) sprintf() 함수

- stdio.h를 선언한다. #include <stdio.h>
- **사용법)** sprintf(변환한 값을 받을 문자열 변수,변환할 정수)
- ex)

```
int cnt=1;
char buffer[10];

sprintf(buffer,"%d",cnt);  // buffer에는 1이 문자열로 변환되어 저장됨.
```

- ex2)

```
int cnt=1;
char buffer[10];

sprintf(buffer,"_%d",cnt);  // buffer에는 _1이 문자열로 저장됨.
```

Sprintf에 "%d" 대신 "0x%x" 하면 16진수로 변환됨(0x는 16진수 구분위함

```
sprintf(uartdata,"%d",adcValue);// int 뿐만 아니라 무슨형이든 ascii 형태로 변환 하는 목적
itoa(adcValue, uartdata, 10); //itoa 와 sprintf에 대해서
```

1. Review-string print

- **UART_transmit(unsigned char data)**
- 일반적인 문자 하나 보내는 출력이다.

- **UART_string_transmit(char *string)**
- **while(*string != '₩0') // null 이 아닐 때까지만 출력하는 것**

```
void UART_transmit(unsigned char data)
{
    while(!(UCSR0A & (1<<UDRE0)));  //wait for empty transmit buffer
    UDR0 = data;    //put data into buffer, sends the data
}

void UART_string_transmit(char *string)
{
    while(*string != '\0')
    {
        UART_transmit(*string);
        string++;
    }
}
```

```
/* Replace with your application code */
char uartdata[2] = {'\0'};
```

2. volatile 조사

- **컴파일러 관점에서 생각해보자**
- 같은 주소에 여러 value 값을 넣는다면 마지막 값만 넣는다
- 반복되거나 자주사용하거나 중요한 변수는 레지스터나 케쉬에 저장한다는점에 주목

- **언제 쓰나?**
- **인터럽트와 외부센서 입력 등과 같은 HW 적으로 값을 변경할 때 적용안될수 있으니 사용**
- **MMIO 에 연결된 SFR 등과 같은 주소에 값을 여러 번 쓰는데 각각이 특정 명령을 전달하는 내용이기 때문에 사용**
- **쓰레드 환경에서 공유하는 전역변수 자원 같은 경우에 사용함**

=> 공유자원에는 전부 volatile 처리를 해야하는 것인가?

```
/* Replace with your application code */
char uartdata[2] = {'\0'};
```

감사합니다.