시스템 프로그래밍을 위한 C언어

Enumeration

현대자동차 입문교육
박대진 교수

Enum 이용한 Bit Manipulation

```
enum color{
    red; /* red is given the value 0 by default */
    blue; /* blue gets the value 1 */
    green; /* green gets the value 2 */
    yellow = 5; /* yellow gets the specified value 5 */
};
```

```
1   enum buttonStates
2   {
3       BUTTON_PULSE_ON,
4       BUTTON_PULSE_OFF,
5       BUTTON_ACTIVE,
6       BUTTON_INACTIVE
7   };
8
9   buttonStates g_buttonState;
10
11
12  void setup ()
13  {
14      g_buttonState = BUTTON_ACTIVE;
15  }
16
```

```
// Enumeration type
typedef enum
{
    LED_RED    =        (1 << 0),
    LED_GREEN  =        (1 << 1),
    LED_YELLOW =        (1 << 2),
    LED_ORANGE =        (1 << 3),
} LedType;

...

// Function declaration
void setOnLed(LedType led);

...

// Function call
setOnLed(LED_RED | LED_GREEN | LED_YELLOW);
```

```
enum ThingFlags = {
  ThingMask  = 0x0000,
  ThingFlag0 = 1 << 0,
  ThingFlag1 = 1 << 1,
  ThingError = 1 << 8,
}
```

Then use the *names* later on. I.e. write

```
thingstate |= ThingFlag1;
thingstate &= ~ThingFlag0;
if (thing & ThingError) {...}
```

Bit Representation using enum

```c
// hello.c
#include <stdio.h>
enum ADC_STATUS { EOC=1, SOC=0 };
enum TIMER_MODE {
    TIMER_EN    = (1<<7),
    TIMER_START = (1<<2)
};
int adc_status() {
    // activating ADC, reading ADC
    // if( read_adc())
    //   return EOC;
    return SOC;
}
```

```c
int main()
{
    if(adc_status() == SOC) {
        printf("still on conversion...\n");
    }

    // #define TMODE *((unsigned char*)0xFFFF0000)
    unsigned char TMODE;
    TMODE  = TIMER_EN;
    TMODE |= TIMER_START;
    printf("TMODE: 0x%2X\n", TMODE);
    return 0;
}
```