**임베디드 소프트웨어 Lab 05 실습 보고서**

**2011104043 컴퓨터공학과 장원엽**

1. 강의 내용

이번 시간에는 OS를 사용한 임베디드 프로그래밍 방법을 배웠다. 우리가 사용할 OS는 uC/OS-2로, 원래는 교육용으로 개발된 OS지만, 뛰어난 성능으로 최근에는 많은 분야에서 사용되고 있는 OS이다.

임베디드 프로그래밍에 사용되는 OS는 Real-time OS라고 불린다. 제한된 시간 내에 계산 결과를 도출해 내야 하기 때문이다. 어떤 작업을 시간 제한에 맞추지 못할 경우 해당 작업은 버려지게 되는데, 이 때문에 시스템에 심각한 오류가 발생하거나 올바르게 작동하지 않을 수 있다.

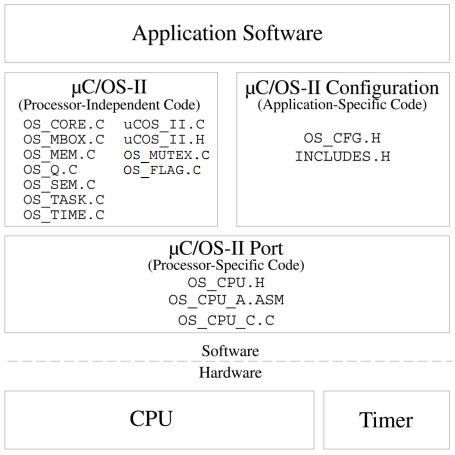
우리가 OS를 사용하는 데에는 몇 가지 이유가 있다. 그 중 하나는 멀티테스킹이다. OS는 여러 가지 작업들을 동시에 수행할 수 있도록 도와준다. 이것을 가능하게 하는 것은 작업들간의 우선순위를 정하고, 세마포어나 뮤텍스를 이용하여 현재 수행중인 작업을 다른 작업들과 동기화 하는 것이다.

작업들간의 우선순위를 정하는 방법 중에서 가장 많이 쓰이는 방법은 Rate-Monotonic Scheduling으로, 이는 가장 짧은 주기로 수행되어야 하는 작업의 우선순위를 가장 높게 설정하는 방법이다.

멀티테스킹을 하기 위해서는 CPU Register에 수행해야 할 각각의 작업 정보를 저장해 놓아야 한다. CPU는 어떤 작업을 수행하다가 다른 작업을 해야 할 필요가 있을 시에는 현재 수행 중이던 작업을 레지스터에 저장한 뒤 다른 작업을 수행하고 다시 레지스터에서 원래 작업 정보를 가져와 원래 작업을 수행한다. 이러한 멀티테스킹 방식을 Preemptive Multitasking이라고 한다.

어떤 작업을 수행 중 ISR에 의하여 다른 작업에 대한 처리 요청이 들어왔을 시, CPU가 작업들간의 우선순위를 판단하고, 현재 수행중인 작업보다 우선순위가 높은 작업일 경우, 현재 수행 중이던 작업을 중단하고 요청된 작업을 수행하는 것이 Preemptive Multitasking이다.

다음은 uC/OS-2를 이루는 구성 요소이다.



uC/OS-2는 위 사진에서 볼 수 있듯이 OS부분, 설정부분, 포트부분으로 나뉜다. OS부분에 uC/OS-2의 핵심 코드들이 들어있다. 코어, 메시지박스, 메모리, 큐, 세마포어, 테스크, 타이머, 뮤텍스, 플레그등의 코드가 포함돼 있는 것을 확인할 수 있다.

1. 실습

#include "includes.h" // uC/OS-2의 설정부분에 포함되어 있는 헤더이다.

#define F\_CPU 16000000UL // CPU frequency = 16 Mhz

#include <avr/io.h>

#include <util/delay.h>

#define TASK\_STK\_SIZE OS\_TASK\_DEF\_STK\_SIZE // 테스크에서 사용할 스텍의 사이즈를 uC/OS-2에서 제공하는 기본 스텍 사이즈로 설정한다.

OS\_STK LedTaskStk[TASK\_STK\_SIZE];

void LedTask(void \*data);

int main (void)

{

OSInit();

OS\_ENTER\_CRITICAL(); // 크리티컬 섹션에 진입하는 코드이다.

TCCR0=0x07;

TIMSK=\_BV(TOIE0);

TCNT0=256-(CPU\_CLOCK\_HZ/OS\_TICKS\_PER\_SEC/ 1024); // CPU의 클럭수를 정의한다.

OS\_EXIT\_CRITICAL(); // 크리티컬 섹션의 작업을 끝낸다.

OSTaskCreate(LedTask, (void \*)0, (void \*)&LedTaskStk[TASK\_STK\_SIZE - 1], 0); // 작업을 생성한다. 첫 번째 변수는 작업의 이름, 두 번째 변수는 작업으로 넘겨줄 인자, 세 번째 변수는 작업에 할당될 스택의 주소, 네 번째 변수는 작업의 우선순위를 의미한다. 0번은 최고 우선순위라는 의미이다.

OSStart(); // OS를 시작한다.

return 0;

}

void LedTask (void \*data) // 작업의 내용을 정의하는 코드이다.

{

data = data;

// OSStatInit();

// Main task

DDRA = 0xff;

while (1) {

PORTA = 0xaa;

//\_delay\_ms(500);

OSTimeDlyHMSM(0, 0, 0, 500); // uC/OS-2에서 제공하는 \_delay\_ms와 같은 기능을 하는 함수이다. 순서대로 시간, 분, 초, 미리초를 의미한다.

PORTA = 0x55;

//\_delay\_ms(500);

OSTimeDlyHMSM(0, 0, 0, 500);

}

}

두 번째 실습에서는 함수 내부의 내용을 FND를 컨트롤 하는 내용으로 바꾸어주기만 하면 된다. FND 컨트롤 코드는 다음과 같다.

unsigned char FND\_DATA[ ]= {0x3f, 0x06, 0x5b, 0x4f, 0x66,

0x6d, 0x7d, 0x27, 0x7f, 0x6f, 0x77, 0x7c, 0x39,

0x5e, 0x79, 0x71, 0x80, 0x40, 0x08};

unsigned int num=0, num0, num1=0, num2=0, num3=0;

unsigned int cnt = 0;

DDRC = 0xff;

DDRG = 0x0f;

while(1)

{

cnt++;

if(cnt == 100){

cnt = 0;

num++;

}

num0 = num%10;

if(num0 == 0 && cnt == 0){

num1++;

}

if(num1 == 6){

num1 = 0;

num2++;

}

if(num2 == 10){

num2 = 0;

num3++;

}

if(num3 == 6){

num3 = 0;

}

PORTC = FND\_DATA[num3];

PORTG = 0x08;

\_delay\_ms(2);

PORTC = FND\_DATA[num2] + 0x80;

PORTG = 0x04;

\_delay\_ms(3);

if(cnt <= 25){

PORTC = FND\_DATA[num1];

PORTG = 0x02;

\_delay\_ms(2);

PORTC = FND\_DATA[num0];

PORTG = 0x01;

\_delay\_ms(3);

}

else if(cnt > 25 && cnt <= 50){}

else if(cnt > 50 && cnt <= 75){

PORTC = FND\_DATA[num1];

PORTG = 0x02;

\_delay\_ms(2);

PORTC = FND\_DATA[num0];

PORTG = 0x01;

\_delay\_ms(3);

}

else{}

}

위 코드는 Lab. 3)의 FND programming 실습을 uC/OS-II task로 구현한 것으로, 실제로 같은 코드를 복사 붙여 넣기 한 것이다.