**임베디드 소프트웨어 Lab 3 실습보고서**

2011104043 컴퓨터공학과 장원엽

1. **강의 내용**

이번 시간에는 GPIO를 이용하여 FND를 제어하는 방법을 배웠다. FND란 Flexible Numeric Device(Display)의 약자로 7 Segment Display가 FND라고 할 수 있다. JKit에서는 이 FND를 제어하기 위해 2개의 포트(C, G)를 제공하고 있다. C포트는 dot(.)을 포함한 7 Segment의 8개의 LED를 제어하기 위한 값을 입력 받으며, G포트는 7 Segment의 4자리 각각을 컨트롤 하기 위한 값을 입력 받는다.

7 Segment를 조절하기 위하여 C포트에 입력하는 값의 자세한 정보는 다음과 같다.



위 사진을 보면 알 수 있듯이, 7 Segment의 LED각각의 자리에 들어갈 값을 0으로 주면 해당 LED의 불이 꺼지고, 1로 주면 해당 LED에 불이 들어온다. 프로그램 상으로 입력 시에는 해당하는 2진값을 16진수로 바꾸어 C포트에 입력해준다.

1. **실습내용**

#include <avr/io.h> // ATmega128 register 정의

void main()

{

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

0x66, 0x6d, 0x7d, 0x27, 0x7f, 0x6f, 0x77, 0x7c, 0x39, 0x5e,

0x79, 0x71, 0x80, 0x40, 0x08}; // 0~9, A~F, ‘.’, ‘-’, ‘\_”

DDRC = 0xff; // C 포트는 모두 출력

DDRG = 0x0f; // G 포트도 4개는 출력

PORTC = FND\_DATA[7]; // C 포트는 FND 데이터 신호

PORTG = 0x01; // G 포트는 FND 선택 신호

}

위 코드는 FND에 7이라는 숫자를 출력하는 코드이다. C포트에 출력 할 16진수 값을 넣어주고, G포트에는 0x01을 입력하여 4자리 7 Segment 중 첫 번째 자리에 출력한다.

#include <avr/io.h>

#include <util/delay.h>

void main()

{

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

0x4f, 0x66, 0x6d, 0x7d, 0x27, 0x7f, 0x6f, 0x77, 0x7c, 0x39, 0x5e,

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

DDRC = 0xff;

DDRG = 0x0f;

while(1)

{

PORTC = FND\_DATA[1];

PORTG = 0x08;

\_delay\_ms(1);

PORTC = FND\_DATA[2];

PORTG = 0x04;

\_delay\_ms(1);

PORTC = FND\_DATA[3];

PORTG = 0x02;

\_delay\_ms(1);

PORTC = FND\_DATA[4];

PORTG = 0x01;

\_delay\_ms(1);

}

}

위 코드는 각각의 7 Segment에 1234를 출력한다. 코드를 보면 while문 안에 FND제어코드를 입력했는데, 이는 하드웨어상의 문제로, 한번에 하나의 7 Segment밖에 제어할 수 없기 때문에 while문을 돌려 각각의 7 Segment를 켜고 끄는 것을 반복하는 것이다.

#define F\_CPU 16000000UL

#include <avr/io.h>

#include <util/delay.h>

void main()

{

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;

DDRC = 0xff;

DDRG = 0x0f;

while(1)

{

num++;

num0 = num%10;

if(num0 == 0){

num1++;

}

if(num1 == 10){

num1 = 0;

num2++;

}

if(num2 == 10){

num2 = 0;

num3++;

}

if(num3 == 10){

num3 = 0;

}

PORTC = FND\_DATA[num3];

PORTG = 0x08;

\_delay\_ms(2);

PORTC = FND\_DATA[num2];

PORTG = 0x04;

\_delay\_ms(3);

PORTC = FND\_DATA[num1];

PORTG = 0x02;

\_delay\_ms(2);

PORTC = FND\_DATA[num0];

PORTG = 0x01;

\_delay\_ms(3);

}

}

위 코드는 1부터 9999까지의 숫자를 세는 코드이다. 첫째 자리에 표시되는 수는 while문이 한번 돌때다마 증가하므로 사람의 눈으로 인식할 수 없으며, 두 번째 자리에 표시되는 수는 첫 번째 자리수가 10까지 증가하면 1 증가한다. 세 번째 자리와 네 번째 자리에 입력되는 수도 이와 같다.

1. **과제**

#define F\_CPU 16000000UL

#include <avr/io.h>

#include <util/delay.h>

void main()

{

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);

}

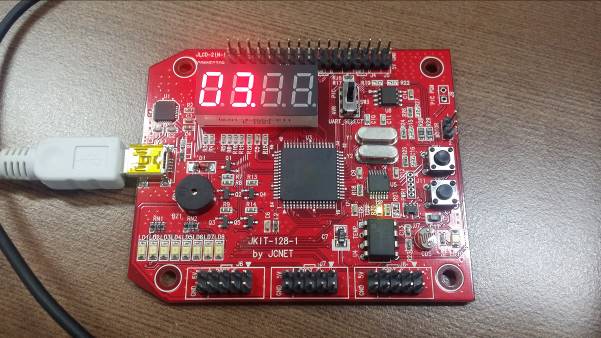
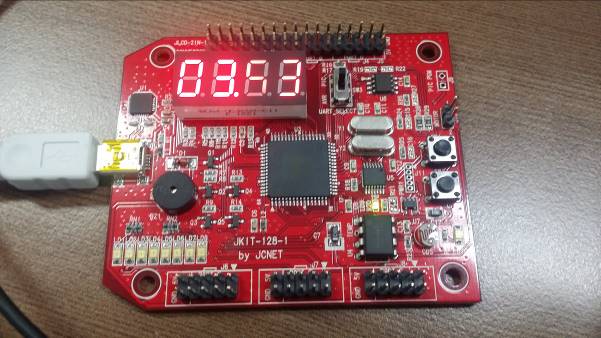
else{}

}

}

위 코드는 1시간까지 셀 수 있는 초 시계이다. 뒤에 두 자리 7 Segment에는 0부터 60까지의 초를 표시하고, 앞에 두 자리 7 Segment에는 0부터 60까지 분을 표시한다. 또한 초를 나타내는 부분이 깜박거리는데, 이는 7 Segment를 표시함에 있어서 \_delay\_ms() 함수를 사용하여 출력 시간과 출력하지 않는 시간에 딜레이를 준 것이다.

1. 결과 사진



뒤에 초를 나타내는 두 자리 숫자는 점멸을 반복하며 60까지 증가하고, 앞의 두 자리 숫자는 분을 나타낸다.