※ 제목: 실험#4 결과보고서 – Step Motor 제어, Timer0 제어

※ 실험일: 2013. 04. 30. (9주차)

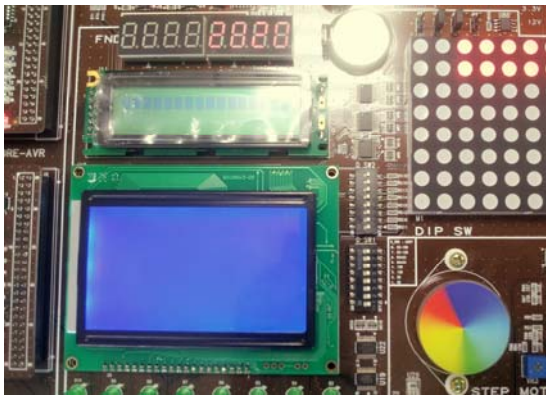
※ 실험 내용

- 실험과제1 : Timer/Counter0을 이용하여, 처음 실행시 1초마다 1회전하도록 스텝모터를 동작시키고, 네 개의 키패드 버튼을 통해 스텝모터의 속도(4, 8번 키) 및 방향(0, 1번 키)을 조절한다. 속도 조절은 2배씩 증가 또는 감소하여 0.25~4초 사이에서 동작하도록 한다.
- 실험과제2 : 위 실험과제1을 Timer/Counter1을 이용하여 동작하도록 수정한다.

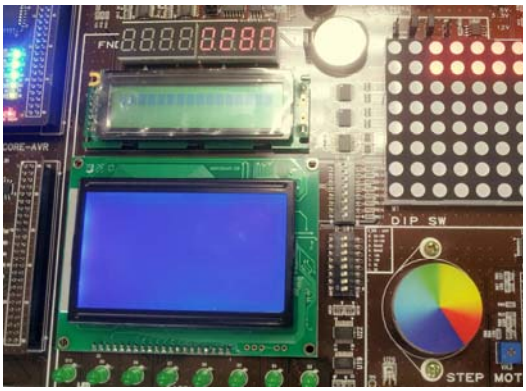
※ 실험 결과

- 실험과제1 : 타이머 0을 통한 스텝모터 제어
  - 해결 방법
    : 타이머 0을 1ms마다 동작하도록 설정한다.
    : 스텝모터의 1회전은 48스텝이기 때문에 1초에 1회전 하려면 1000 / 48 ≒ 20ms마다 다음 시퀀스를 출력해야한다. 따라서 1ms마다 동작하는 타이머의 반복 횟수를 저장하는 변수를 통해 초기값을 20번 반복시 마다 다음 시퀀스를 제어하도록 작성하고, 키 입력에 따라 반복횟수를 증감시키도록 작성하였다.
    : 또, 현재 방향값을 저장하는 변수를 두어, 키 입력에 따라 이 변수를 반전시키고, 타이머 동작시 이 변수의 값에 따라 시퀀스 번호의 증감을 설정하도록 작성하였다.
    : 7-Segment에 회전수와 현재 딜레이를 표시하여 확인하기 쉽도록 작성하였다.

  - 결과
    : 키패드의 버튼 조작으로 스텝모터의 속도 및 회전 방향 제어가 원활하게 동작되는 것을 확인하였다.



- 1초에 1회전중인 스텝모터 -



- 4초에 1회전중인 스텝모터 -

- 소스코드

```
#include <avr/io.h>

#define STEP_CS    (*(volatile unsigned char *)0x4000)
#define KEY_DATA   (*(volatile unsigned char *) 0x5400)
#define KEY_SCAN   (*(volatile unsigned char *) 0x5800)
#define FND_DATA (*(volatile unsigned char *) 0x6C00)
#define FND_COM   (*(volatile unsigned char *) 0x7000)

volatile uint8_t StepMotorCnt, dir, TCNT_CNT, TCNT_CNT_DEF;
volatile uint8_t RotationCnt, StepCnt;

void Timer0(void) {
  TCNT_CNT--;
  if (!TCNT_CNT) {
    if (dir)
      StepMotorCnt = (StepMotorCnt == 4) ? 1: ++StepMotorCnt;
```

```
메모리맵 IO를 위해 외부 LED의 매핑 주소 정의

스텝시퀀스, 회전방향, 반복 카운트, 반복 카운트 초기화 값
회전 수 카운트, 스텝 수 카운트

타이머0 이벤트 처리 함수.
반복 횟수 카운터 값을 감소시킨다.
지정한 수 만큼 반복되어 카운터 값이 0이 되었으면
현재 회전 방향이 반시계방향이면 시퀀스 번호를
증가시키고, 시계방향이면 시퀀스 번호를 감소시킨
```

```
    else
      StepMotorCnt = (StepMotorCnt == 1) ? 4: --StepMotorCnt;
    StepMotorForward();
    TCNT_CNT = TCNT_CNT_DEF;
  }
  TCNT0 = 0;
  TIFR = 0b00000001;
}

void Timer0_Init(unsigned char delay) {
  TCNT_CNT_DEF = delay;
  TCNT_CNT = TCNT_CNT_DEF;
  TCCR0 = 0x03;
  ASSR = 0x00;
  TCNT0 = 0;


  TIMSK = 0x00;
  TIFR = 0b00000001;
  SREG &= 0x7F;
}

void StepMotorForward(void) {
  switch(StepMotorCnt) {
  case 1: STEP_CS = 0x01; break;
  case 2: STEP_CS = 0x04; break;
  case 3: STEP_CS = 0x02; break;
  case 4: STEP_CS = 0x08; break;
  }

  StepCnt++;
  if (StepCnt > 47) {
    StepCnt = 0;
    RotationCnt++;
  }
  if (RotationCnt > 99) RotationCnt = 0;
}

unsigned char getKey(unsigned char row) {
  unsigned char i, ks, val = 0xFF;

  KEY_DATA = ~(0x1 << row);
  ks = ~KEY_SCAN & 0x0F;
  for (i = 0; i < 4; i++)
    if (ks & (0x1 << i)) val = (3 - i) + (row * 4);

  return val;
}

unsigned char Num2Seg(unsigned char n) {
  unsigned char seg = 0;

  switch (n) {
    case 0x0: seg = 0x3F; break;
    case 0x1: seg = 0x06; break;
    case 0x2: seg = 0x5B; break;
    case 0x3: seg = 0x4F; break;
    case 0x4: seg = 0x66; break;
    case 0x5: seg = 0x6D; break;
    case 0x6: seg = 0x7D; break;
    case 0x7: seg = 0x07; break;
    case 0x8: seg = 0x7F; break;
    case 0x9: seg = 0x6F; break;
  }

  return ~seg;
}
```

다.

시퀀스 번호에 따라 스텝모터에 신호를 출력한다.
반복 카운트 변수를 다시 초기화한다.

타이머 0의 TCNT값을 0으로 초기화하고, TOV0플래그를 0으로 만들어 타이머가 계속 동작하도록 한다.

타이머0 초기화함수
인수로 받은 delay를 카운트 초기화값으로 설정하고, 현재 카운트 값도 같이 초기화한다.
타이머0을 32분주비로 동작하도록 설정한다.
타이머 0를 동기방식으로 작동하도록 설정한다.
타이머가 0~255까지 256클럭마다 동작하도록 TCNT값을 0으로 초기화한다.
따라서 0.125 * 32 * 256 = 1024us ≒ 1ms가 된다.
타이머 인터럽트를 사용하지 않는다.
TOV0 플래그를 0으로 초기화한다.
SREG 레지스터의 I비트를 0으로 만들어 인터럽트를 사용하지 않도록 설정한다.

시퀀스 번호에 따라 스텝모터 제어신호를 출력하는 함수
7주차 스텝모터 제어 참조.

스텝 카운터를 증가시킨다.
48회전시 스텝 카운터를 0으로 초기화하고, 회전수 카운터를 증가시킨다.

7세그먼트 두자리를 사용하기 때문에 회전수 카운터는 0~99값만을 사용한다.

키스캔 결과값 반환 함수
7주차 스텝모터 제어 참조.

인수로 받은 숫자에 해당하는 7세그먼트 값을 반환하는 함수. 6주차 7세그먼트 제어 참조.

```
void showSeg(unsigned char com, unsigned char data) {
  FND_DATA = 0xFF;
  FND_COM = com;
  FND_DATA = data;
}

void showSegDec(unsigned char com, unsigned char digit,
unsigned char num) {
  unsigned char i;

  for (i = 0; i < digit; i++) {
    showSeg(com >> i, Num2Seg(num % 10));
    num = num / 10;
  }
}

int main(void) {
  unsigned char row = 0, val = 0xFF, val_pre = 0xFF;
  unsigned char delay = 20;

  MCUCR = 0x80;
  StepMotorCnt = 1;
  RotationCnt = 0;
  StepCnt = 0;
  StepMotorForward();
  Timer0_Init(delay);

  while (1) {
    showSegDec(0x20, 2, RotationCnt);
    showSegDec(0x80, 2, TCNT_CNT_DEF);
    val = getKey(row);

    if (val != 0xFF && val != val_pre) {
      switch (val) {
      case 0: dir = 0; break;
      case 1: dir = 1; break;
      case 4:
        delay = (delay > 5) ? delay / 2: delay; break;
      case 8:
        delay = (delay < 80) ? delay * 2: delay; break;
      }

      RotationCnt = 0;
      StepCnt = 0;
      Timer0_Init(delay);
    }
    if (TIFR & 0b00000001)
      Timer0();
    if (val == 0xFF) row = (row + 1) % 4;


    val_pre = val;
  }
  return 0;
}
```

출력 할 FND의 열과, 세그먼트 값을 인수로 받아 7세그먼트에 출력하는 함수.
6주차 7세그먼트 제어 참조.

출력을 시작 할 FND의 열과, 출력 할 자릿수, 출력 할 수를 인수로 받아 10진수 숫자를 FND에 출력하는 함수.
6주차 7세그먼트 제어 참조.

타이머 반복횟수 카운트의 초기값 = 20 (20ms)

외부메모리 IO를 위해 SRE비트를 1로 설정
스텝모터 시퀀스 번호를 1로 초기화
회전수, 스텝수를 0으로 초기화

스텝모터를 한스텝 이동시킨다.
현재 값으로 타이머를 초기화한다.

현재 회전수와 딜레이를 7세그먼트에 출력한다.

눌려진 키를 조사한다.

눌러진 버튼이 있고, 키를 계속 누르고 있는 경우가 아니라면, 키 값에 따라 동작을 수행한다.
0번키: 반시계방향 회전
1번키: 시계방향 회전
4번키: 속도 증가
  속도를 4회전/1초 이하로 제한
8번키: 속도 감소
  속도를 1회전/4초 이상으로 제한

동작이 변경되었기 때문에 다시 초기화한다.
회전수 초기화
스텝수 초기화
위에서 계산된 딜레이로 타이머 초기화

TCNT가 255이후 0이 되어 TOV0 플래그가 1이 되었다면 타이머0 이벤트 처리 함수를 호출한다.
키 입력이 없을 때만 각 행을 순차적으로 스캔하고, 키 입력이 있을 때엔 스캔값을 현재 행으로 고정한다.
이전 버튼값 저장 변수를 갱신한다.

- 실험과제2 : 타이머 1을 통한 스텝모터 제어
  - 해결 방법
    : 실험과제1의 타이머 초기화 관련 레지스터를 타이머1의 레지스터로 변경하고, TOV0이 아닌 TOV1의 상태를 조사하도록 수정하였다.
    : 타이머1은 32분주비를 사용할 수 없기 때문에 64분주비를 사용하고, 65407~65535까지 128클럭, 즉 타이머0의 코드의 절반만 사용하도록 하여 1ms를 구현하였다.

  - 결과
    : 실험과제1의 결과와 동일하게 동작하는 것을 확인하였다.

- 소스코드

	실험과제1의 소스코드 중 Timer0과 Timer0_Init함수만을 재작성하였다.

void Timer0(void) {
  TCNT_CNT--;
  if (!TCNT_CNT) {
  if (dir)
  StepMotorCnt = (StepMotorCnt == 4) ? 1: ++StepMotorCnt;
  else
  StepMotorCnt = (StepMotorCnt == 1) ? 4: --StepMotorCnt;
 	
  StepMotorForward();
  TCNT_CNT = TCNT_CNT_DEF;
  }
 	
  TCNT1H = 0xFF;
  TCNT1L = 0x7F;
  TIFR = 0b00000100;
}
	타이머 이벤트 처리 함수.
	반복 횟수 카운터 값을 감소시킨다.
	지정한 수 만큼 반복되어 카운터 값이 0이 되었으면 현재 회전 방향이 반시계방향이면 시퀀스 번호를 증가시키고, 시계방향이면 시퀀스 번호를 감소시킨다.
	
	시퀀스 번호에 따라 스텝모터에 신호를 출력한다.
	반복 카운트 변수를 다시 초기화한다.
	
	타이머 1의 TCNT값을 다시 65407로 초기화하고, TOV1 플래그를 0으로 만들어 타이머가 계속 동작하도록 한다.

void Timer0_Init(unsigned char delay) {
  TCNT_CNT_DEF = delay;
  TCNT_CNT = TCNT_CNT_DEF;
  TCCR1A = 0x00;
  TCCR1B = 0x03;
  TCCR1C = 0x00;
  TCNT1H = 0xFF;
  TCNT1L = 0x7F;
  TIMSK = 0x00;
  TIFR = 0b00000100;
  SREG &= 0x7F;
 	
  RotationCnt = 0;
  StepCnt = 0;
}
	타이머 초기화함수
	인수로 받은 delay를 카운트 초기화값으로 설정하고, 현재 카운트 값도 같이 초기화한다.
	타이머1을 64분주비로 동작하도록 설정한다.
	
	타이머가 65407~65535까지 128클럭마다 동작하도록 TCNT1값을 65407로 초기화한다.
	타이머 인터럽트를 사용하지 않는다.
	TOV1 플래그를 0으로 초기화한다.
	SREG 레지스터의 I비트를 0으로 만들어 인터럽트를 사용하지 않도록 설정한다.
	
	회전수와 스텝 수 카운트를 초기화한다.

※ 실험 후 느낀점

- 프로그램 실행 중에 타이머로 인해 만들어지는 딜레이를 어떻게 가변적으로 제어 할 수 있는지 알 수 있었다.
- 또, 7세그먼트를 동시에 조작하기 위해 변경되는 값만을 타이머 동작함수에서 처리하고, 7세그먼트 출력은 메인 함수에서 처리하는 등, 다수의 소자를 제어할 때의 응용 방법을 알게 되어 흥미로웠다.