

# 목 차

## 제 1 장 KD-128PRO(ATmega128 키트) 회로도 강좌

---

|                                                   |  |
|---------------------------------------------------|--|
| 1. KD-128PRO-C 키트 회로 구성 개념 .....                  |  |
| 2. 전원부 회로 구성 .....                                |  |
| 2.1 DC 5V 만들기 .....                               |  |
| 2.2 전원부 연결 .....                                  |  |
| 3. 클럭부 연결 .....                                   |  |
| 3.1 ATMEGA128에 연결 가능한 클럭 소스들 .....                |  |
| 3.2 외부 크리스탈 사용시 .....                             |  |
| 3.3 클럭부 연결 .....                                  |  |
| 3.4 PCB에서의 클럭부 설계의 예 .....                        |  |
| 3.5 ISP 회로에 관하여 .....                             |  |
| 3.6 ISP 회로의 보완에 관하여 .....                         |  |
| 3.7 리셋 회로의 연결 .....                               |  |
| 3.8 리셋 회로의 보완 .....                               |  |
| 3.9 ATMEGA128의 SRAM Data 메모리 맵 .....              |  |
| 3.10 SRAM을 AVR에 확장하는 방법 .....                     |  |
| 3.11 KT-M128 키트에서의 Memory Mapped I/O의 회로 구성 ..... |  |
| 3.12 LED 부분 회로도와 소스코드의 작성 .....                   |  |
| 3.13 스위치 부분 회로도와 소스코드의 작성 .....                   |  |
| 3.14 세븐 세그먼트 부분 회로도와 소스코드의 작성 .....               |  |

## 제 1 장. KD-128PRO(ATmega128 키트) 회로도 강좌

(주) 뉴티씨 ( NewTC )

<http://www.NewTC.co.kr>

### 1 KD-128PRO 키트 회로 구성 개념

Memory mapped I/O를 이용하여, 외부 램 영역에 I/O 들이 맵핑되어 있어서, 외부 램을 사용하는 것처럼, 하드웨어를 구성하고, 소프트웨어를 짜서 I/O를 제어할 수 있습니다. 메모리는 프로그램 메모리와 데이터 메모리, EEPROM 메모리가 있으며, 프로그램 영역으로 128KB의 플래시 메모리를 내장하고 있다.

ATmega128의 모든 명령어가 16비트 또는 32비트로 되어 있어, **프로그램 메모리**는 플래시 메모리 영역을 말하며, 그 크기는 64K x 16비트(64x16=1M비트 또는128K 바이트 또는 64k 워드)이다.

**데이터 메모리**는 램 영역을 말하며, 일반 모드일 경우 4096바이트(4096 x 8 비트)크기이며, 외부 확장 메모리를 지원하고 있으나 실습보드에서는 외부 확장 메모리를 사용하지 않는다. 데이터 메모리에서는 레지스터 파일, I/O 레지스터 등이 저장된다.

각 포트의 입력/출력을 테스트 할 수 있도록 포트별로 10핀 헤더박스를 이용하여 (8핀의 데이터비트와 2핀의 전원(5V, GND) 핀으로 구성) 포트를 뽑아 두었으며, 128 모듈은 고장이 나면, 쉽게 교체 가능하도록 착탈식으로 제작하였다.

또한, UART 커넥터의 경우 Molex 5267-3핀 호환커넥터로 RS-232C 레벨(+12V)을 두 포트 사용가능하고, Molex 5045-4핀 호환 커넥터로 UART 레벨(5V TTL Level/ 3.3V CMOS Level 호환 )을 두 포트 사용 가능합니다. 이 두 가지 레벨에서 동시에 선을 뽑아서 사용해도, 사용이 가능하도록, 만들어져 있으므로, 두 레벨에 각각 장치를 연결하여 사용 가능합니다.

동작확인용 LED 2기와 스위치 2기, 또한 4개의 세븐세그먼트 모듈을 통하여 기본적인 입출력 실습이 가능하고, RTC인 1307 칩의 내장으로 I2C(내장은 TWI 포트)를 사용한 실습이 가능합니다.

요즘, 임베디드를 공부하는 분들이 소프트웨어에만 집중하고, 하드웨어는 등한시하는 경향이 있어서, 본 강좌를 작성하게 되었습니다. 부디, 본 강좌를 읽고서, AVR에 관해서는 하드웨어에도 강한 엔지니어가 되었으면 좋겠습니다.

## 2 전원부 회로 구성

### 2.1 DC 5V 만들기

아답타로부터 받은 9~28V DC 전압을 레귤레이터인 7805를 거쳐서 5V를 만듭니다. 세라믹 커패시터 104(0.1uF), 전해 커패시터 100uF를 이용하여 각각 고주파 및 저주파 노이즈를 제거합니다.

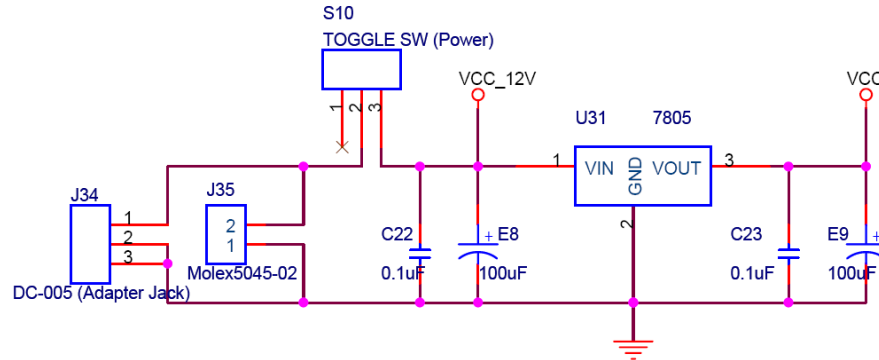
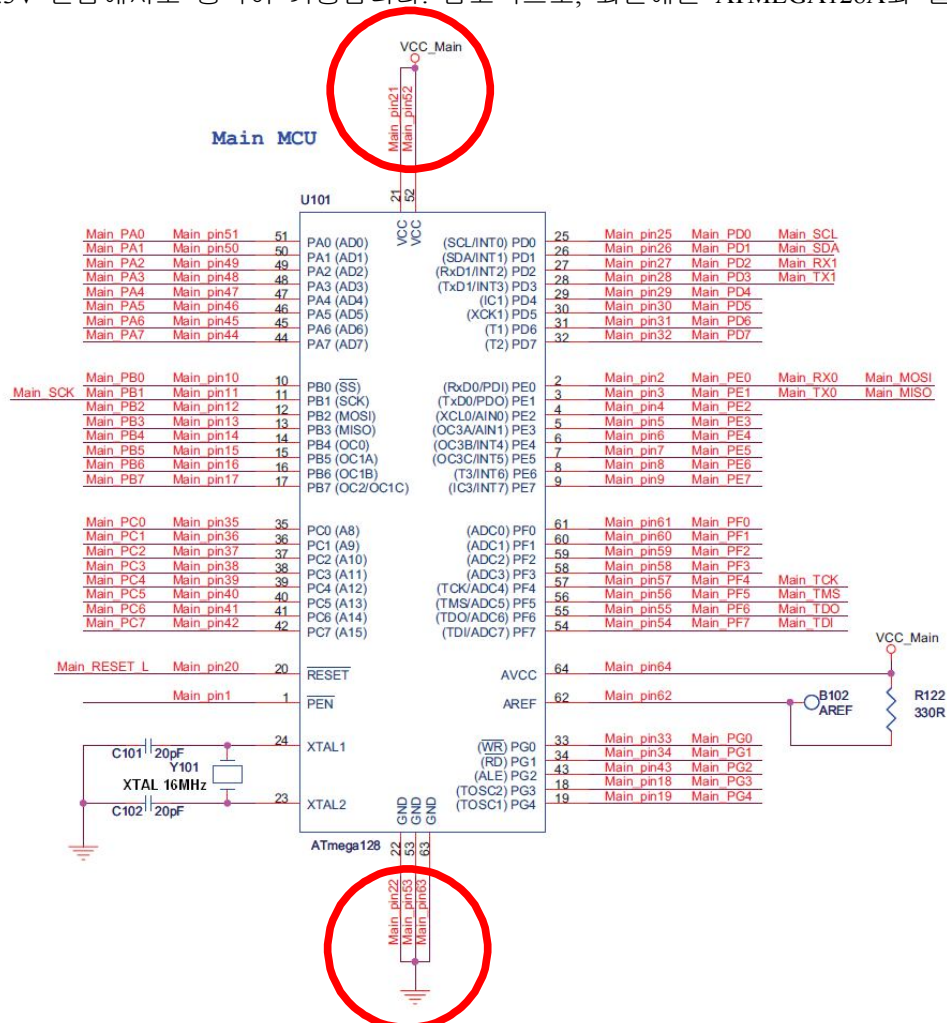


그림 1 간단한 전원부 회로 만들기

### 2.2 전원부 연결

VCC에 5V를 연결하고, GND를 전원부와 연결합니다. 이 회로에서 사용하는 AVR의 경우 TTL 타입으로 5V를 전원으로 사용하지만, ATMEGA128L 이라고 하는 Low voltage CMOS 타입의 경우는 3.3V 전압에서도 동작이 가능합니다. 참고적으로, 최근에는 ATMEGA128A와 같은 A가



붙는 타입이 출시되었는 데, 이는 3.3 / 5V 어떤 전압이든 대응이 가능한 구조로 만들어져 있으며, 이 타입의 경우 5V를 VCC로 사용하면 16MHz까지, 3.3V를 VCC로 사용하면 8MHz까지 클럭 사용이 가능합니다.

RS-232C와 같은 통신을 할 경우, 에러율을 0%로 만들기 위해서는 아래와 같이 크리스탈 주파수를 다른 것으로 변경해 주는 것이 좋습니다. 그렇게 하면, 시리얼 통신 에러가 0%로 떨어지게 되어서, 과도현상으로 인하여 통신 에러율이 높아지는 시스템 리셋이나 새로 전원 인가 시 등에도 에러없는 통신이 가능합니다.

a) 5V VCC를 사용하는 경우 : 14.7458 MHz 또는 11.0592 MHz

b) 3.3V VCC를 사용하는 경우 : 7.3728MHz

### 3 클럭부 연결

#### 3.1 ATMEGA128에 연결 가능한 클럭 소스들

아래와 같이 퓨즈비트 변경에 의하여 선택 가능한 클럭 소스 옵션들이 있으며, 설계시 선택된 소스로부터의 클럭이 AVR 클럭 생성기에 입력되고, 적절한 모듈들의 클럭 근원이 됩니다.

| 디바이스에 클럭을 주는 옵션   | CKSEL3:0    |
|-------------------|-------------|
| 외부 크리스탈 / 세라믹 공진기 | 1111 - 1010 |
| 외부 저주파 크리스탈       | 1001        |
| 외부 RC 오실레이터       | 1000 - 0101 |
| 보정된 내부 RC 오실레이터   | 0100 - 0001 |
| 외부 클럭             | 0000        |

표 2 AVR의 클럭 소스들의 설정법

위에서 하나를 선택하여 사용하며, 처음 AVR을 땀하여 사용할 때에는 CKSEL="0001", 그리고 SUT = "10"으로 선택되어 있으므로, 가장 긴 시작시간을 가진 내부 RC 오실레이터입니다. 그러므로, 정상적인 동작을 위하여 처음 땀한 AVR에는 반드시 퓨즈비트를 변경하여 사용하는 클럭으로 변경하여 주어야 합니다. KD-128PRO-C 키트에서는 외부 크리스탈을 사용하므로, 이 값을 공장출하시에 CKSEL="1111"로 변경하여 두었습니다.

#### 3.2 외부 크리스탈 사용시

XTAL1, XTAL2가 입력과 출력으로 사용되며, 수정 크리스탈 또는 세라믹 공진기 둘다 사용이 가능합니다. CKOPT 퓨즈는 두 개의 다른 오실레이터 앰프 모드들 사이에 선택되는 데, CKOPT가 설정되면, 오실레이터 출력은 출력 상에 풀로 스윙할 것이며, 매우 노이즈가 많은 환경이나 XTAL2의 출력이 다음 클럭 버퍼를 드라이브 할 때 적합하며, 매우 넓은 주파수 범위를 가지고 있습니다. CKOPT가 설정되지 않으면, 오실레이터는 좀 더 작은 출력 스윙을 가지며, 꽤 전원 소모를 감소시킵니다. 이 모드는 제한된 주파수 범위를 가지며, 다른 클럭 버퍼들을 드라이브할 때 사용될 수 없습니다. C1과 C2는 크리스탈과 공진기 모두에서 같은 값이어야 하며, 커패시터들의 이상적인 값은 사용 중인 크리스탈 또는 공진기, PCB 및 실제 마운팅 되는 커패시턴스, 크리스탈에 들어 있는 커패시턴스 등을 합한 부유커패시터의 양, 환경의 전자기파 노이즈 등에 의존합니다. 크리스탈들에 사용할 기본적인 커패시터를 고르는 가이드 라인들이 주어지는 데, 외부 크리스탈에 관해서는 반드시 0.9MHz 이상을 사용하고, 커패시터는 12pF - 22pF가 추천되며, 세라믹 공진기에 대해서는 0.4MHz 이상이 사용 가능하며, 제조자에 의해 선택된 커패시터 값들을 사용하여야 합니다.

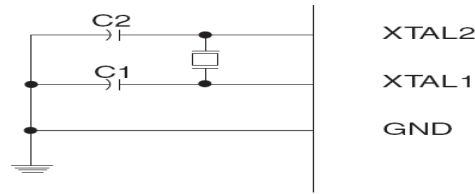


그림 5 클럭부 연결

### 3.3 클럭부 연결

클럭부 소자 연결 시 반드시 준수해야 하는 몇 가지 주의사항이 있습니다. 일단, 크리스탈은 모든 MCU 회로의 기본 클럭이 되므로, 이 부분에 노이즈가 생기거나 타게 되면, 모든 회로에 노이즈가 났게 된다는 것을 의미하므로, 정확한 시스템 동작을 위한 설계에서는 가능한 짧은 선으로 AVR의 핀에 연결하여야 합니다. 또한, 여기서 사용하는 커패시터들도 마찬가지로 가능한 짧은 선을 이용하여야 합니다. PCB 설계 시에는 LGP(Localized Ground Plane)으로 설계하며, 다른 선들이 클럭부 주변을 지나가지 않도록 배선하여야 합니다. 주변으로 지나갈 경우, 크리스탈에서 발생한 신호가 크로스토크를 일으켜서, 다른 선의 신호에 영향을 주거나, 그 선의 데이터가 크리스탈 신호 자체에 영향을 줘서 순간 시스템의 클럭이 무너지거나 영향을 받을 수도 있으므로, 매우 주의하여야 합니다.

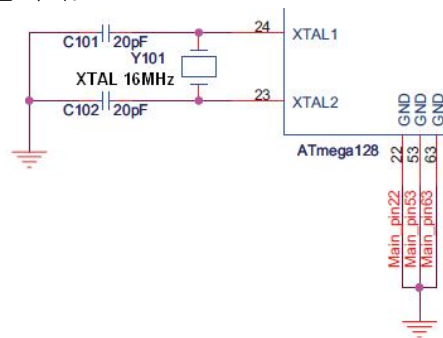


그림 6 ATMEGA128의 클럭부 설계

### 3.4 PCB에서의 클럭부 설계의 예

클럭부 설계 시 아래 그림들과 같이 LGP(Local Ground Plane, 이하 LGP로 표기)를 생성하고, 선 길이는 최소화 하며, 다른 선들이 주변으로 지나다니지 않도록 합니다.

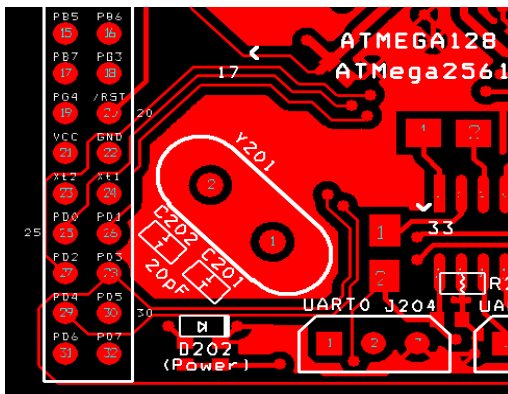


그림 7 클럭부 PCB 설계의 예(BOT면)

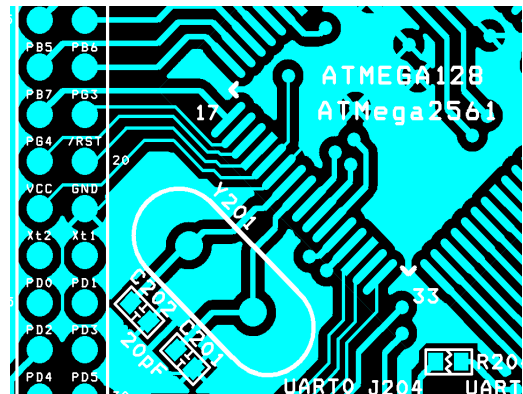


그림 8 클럭부 PCB 설계의 예(TOP면)

### 3.5 ISP 회로에 관하여

ATMEL의 AVR은 USB 방식의 USBISP나 패러렐 방식의 ISPPRO와 같은 상용 비 휘발성 메모리 프로그래머 또는 AVR 코어 상에 동작되는 온칩 부트 프로그램에 의하여 온칩 ISP 플래쉬에 SPI 시리얼 인터페이스를 통하여 시스템의 프로그램 메모리로 재프로그래밍될 수 있도록 되어 있습니다. 플래쉬와 EEPROM 메모리 배열 둘 다 /RESET핀이 GND로 당겨진 동안 시리얼 SPI 버스를 이용하여 프로그램 될 수 있습니다. 시리얼 인터페이스는 SCK, MOSI(입력), MISO(출력)으로 구성되며, /RESET이 0이 된 후에, 프로그래밍 인가 명령어가 프로그램/지우기 명령들이 수행되기 전에 수행되는 것이 필요합니다.

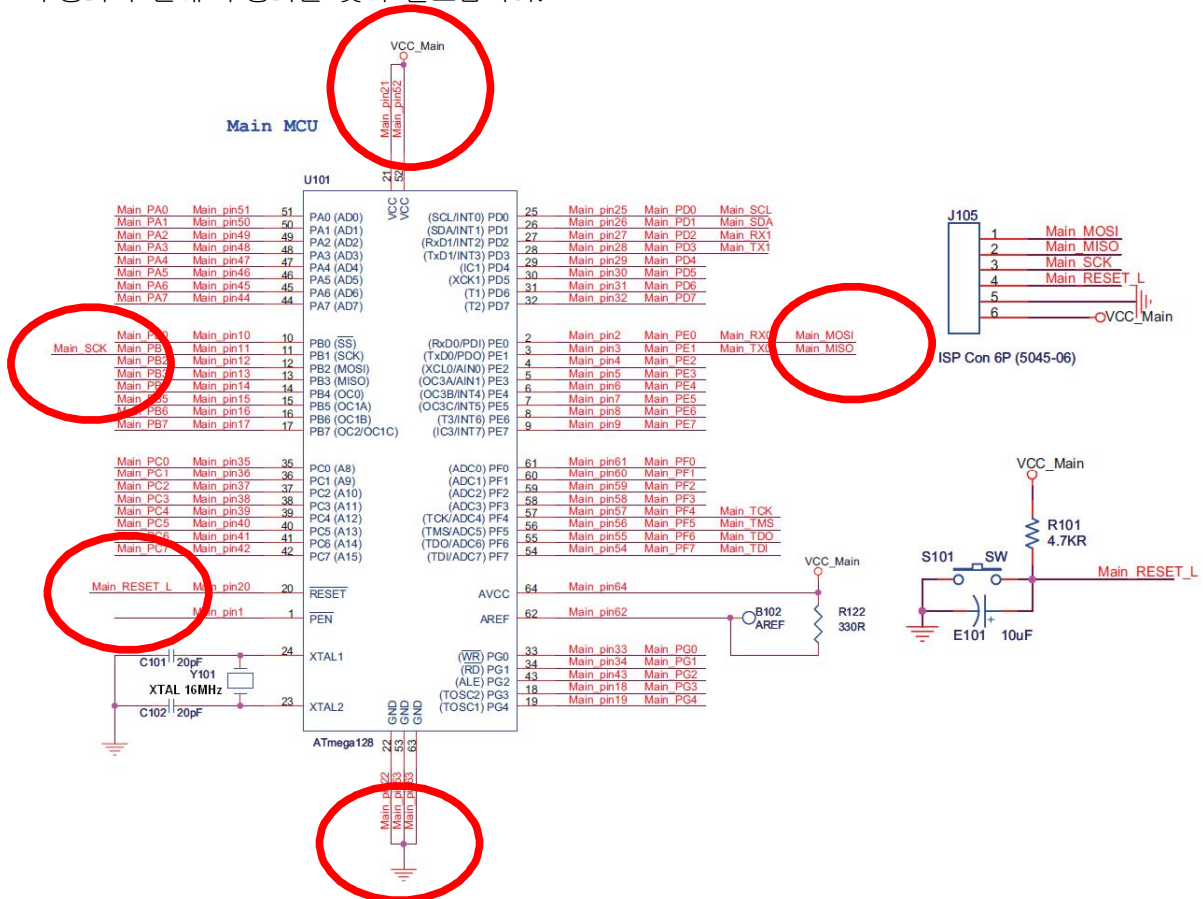


그림 14 ATMEGA128에서의 ISP 회로

다만, 모든 파트들이 내부 SPI 인터페이스를 위하여 전용의 SPI 핀들을 사용하지는 않는다는 점을 주목해 주시기 바랍니다. 예를 들어서, ATMEGA128은 이들 핀들이 PDI와 PDO에 매핑되어 있습니다. ISP 회로 중에서 특히, SCK 라인은 노이즈에 민감하니, PCB 라우팅 시에 유의하여 라우팅 하여야 합니다. 회로 구성상에 SCK 라인 등이 길어진다면, GND 등으로 보호 트레이스를 만들어 노이즈가 끼어드는 것을 막는 것이 좋습니다.

### 3.6 ISP 회로의 보완에 관하여

보통 일반 칩들이 다운로드 핀은 일반 핀들과 별도로 구분되어 사용되어 별 문제가 없지만, AVR과 같은 칩은 핀 수를 줄이기 위해서, 여러 가지 기능을 복합적으로 한 핀에 사용하다 보니, 외부에 붙는 칩들에 영향을 주는 경우가 있습니다. 예를 들자면, 외부에 AVR의 SPI를 사용하는 디바이스가 있다고 할 경우, AVR에 프로그램 라이트 하려고 했는 데, 주위 디바이스까지 신호가 인가되어, 칩의 내용이 변경되는 경우가 있어서, 심각한 피해를 주는 경우가 있습니다. 램과 같은 것이었다면, 큰 피해는 없겠지만, OTP 전용 칩이었다면, 문제는 달라집니다. OTP의 내용이 의도하지 않게 변경되어, 다시는 라이트 못하게 되면서, 심지어는 칩이 망가지는 경우입니다. 이런 경우, 칩을 떼어내고 다시 장착해야 하는 경우까지 생깁니다. 이 경우 아래 회로를 이용하면, 문제없이 원인을 제거할 수 있습니다.

또한, ATMEGA128의 경우는 MOSI/MISO 대신에 사용되는 PDI/PDO가 각각 UART와 공용으로 사용되고 있는 데, 프로그램 라이트 할 때마다, 시리얼 포트로 쓰레기 데이터가 유입되어 시리얼 통신 프로그램이 먹통이 되는 경우가 종종 생기게 됩니다. 이러한 경우도, 아래 회로로 막아낼 수 있습니다.

동작 원리를 보자면, MCU가 플래시 라이트를 하는 경우, RST 핀이 GND로 당겨지게 되는 데, 이 경우는 X, Y, Z에 연결된 AVR의 MOSI/MISO/SCK 라인이 각각 X0, Y0, Z0로 연결되어, 프로그램 라이트에만 영향을 미치게 됩니다. 하지만, 평상시에는 RST 핀이 High로 물려 있으면서 MCU의 MOSI/MISO/SCK 라인이 각각 X1, Y1, Z1으로 물려 있게 되면서, I/O에만 관계되게 됩니다. 매우 간단하면서도, 훌륭하게 위의 어려움을 해결하는 회로이지요. 알아두면, 상용의 시스템을 설계하는 경우 매우 유용할 것으로 생각되어, 공개하니 참고 바랍니다. 이는 본사에서 직접 제작하여 이미 검증하여 사용중인 회로입니다.

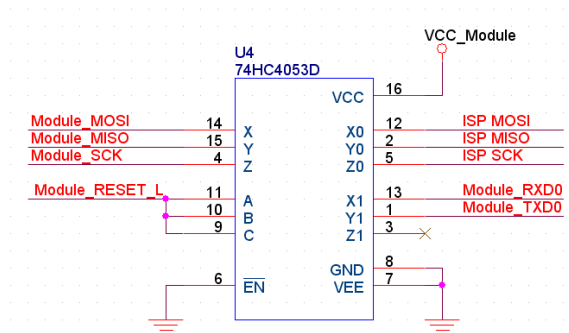


그림 15 ISP 회로의 보완책

### 3.7 리셋 회로의 연결

일반적으로는 아래 그림과 같이 간단하게는 리셋선을  $4.7k\Omega \sim 10k\Omega$ 으로 VCC로 풀업해 주고,  $1 \sim 10\mu F$  정도로 GND에 물려줍니다. 필자는  $4.7k\Omega$ 과  $1\mu F$ 를 여러 시스템에서 테스트 결과 가장 안정적인 값으로 선호하고 있습니다. AVR은 모두 /RST로 Active Low입니다. 이는 8051의 리셋과는 반대임에 유의바랍니다.

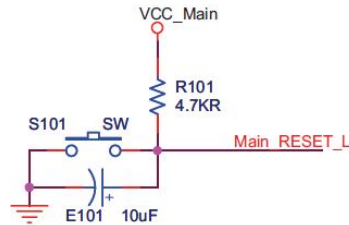


그림 16 기본적 리셋 구성

### 3.8 리셋 회로의 보완

AVR에서 리셋 회로는 Active Low로 최소  $2\mu s$  정도는 low로 유지가 되어야 정상 동작합니다. 사람의 손에 의한 수동 동작이나, RC를 통한 타우값 계산으로 만들어낸 리셋 회로는 최소 동작 시간은 물론, 전압도 순간이긴 하지만, 확대하여 보면 서서히 아날로그적으로 상승하기 때문에 POR(Power On Reset)의 경우 위의 기본 회로만으로는 상용 회로에서는 항상 안정적인 리셋이라고 볼 수 없습니다. 따라서, 상용 회로에서는 안정적인 리셋 회로의 구성이 필수적인데, 이를 위하여 리셋 전압과 리셋 유지 시간에 따른 리셋전용 IC를 사용하거나, 인버터 회로와 버퍼 등을 이용하여, 클럭 파형이 보다 정확히 리셋으로 입력되면서 리셋 유지 시간도 원하는 시간 만큼 될 수 있도록 유지시켜 주는 것이 필요합니다. 이를 위해서, 아래와 같은 몇 가지 회로를 추천합니다.

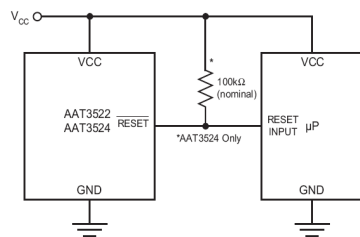


그림 17 5V용 리셋회로 보완

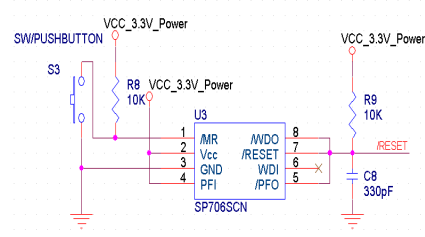


그림 18 3.3V용 리셋회로 보완

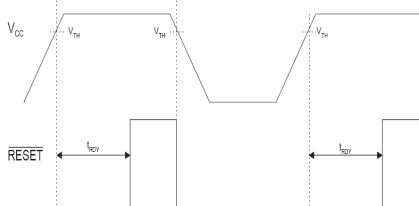


그림 19 5V용 리셋 타이밍도

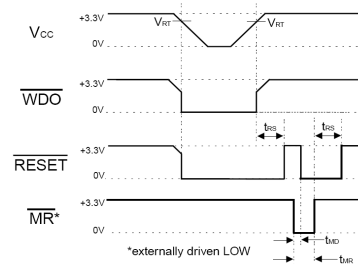


그림 20 3.3V용 리셋타이밍도

### 3.9 ATMEGA128의 SRAM Data 메모리 맵

ATMEGA128에서는 ATMEGA128모드와 ATMEGA103 모드의 퓨즈비트 셋팅을 이용하여 마치 두 가지 MCU인 것처럼 동작이 가능한데, ATMEGA128 모드로 사용하여야 칩 전체 기능을 이용할 수 있어서, 보통은 ATMEGA128 모드로 사용합니다. 이럴 경우, Memory Configuration A로 동작하며, 이는 아래와 같습니다.

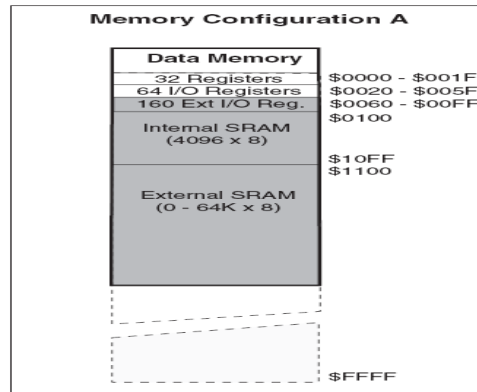


그림 21 Memory Configuration A

여기서 보면 외부 SRAM 영역이 \$1100~\$FFFF까지 사용할 수 있는 것으로 나옵니다. 이 영역을 SRAM을 장착하여, 실제로 메모리를 늘려서 사용할 수도 있겠고, 이 공간을 AVR이 메모리처럼 I/O를 쓰도록 이 영역에 I/O를 하드웨어적으로 물려 놓으면, Memory mapped I/O로 동작하게 됩니다.

### 3.10 SRAM을 AVR에 확장하는 방법

ATMEGA128에서 외부 메모리 인터페이스를 사용하는 것은 실제로는 매우 간단합니다. 이 때 인터페이스 회로는 아래와 같이 구성됩니다.

- 아래 -

AD7:0 : 멀티플렉스된 하위 8비트 어드레스와 데이터버스

A15:8 : 상위 8비트 어드레스 버스

ALE : 어드레스 래치 인에이블

/RD : 읽기 스트로브

/WR : 쓰기 스트로브

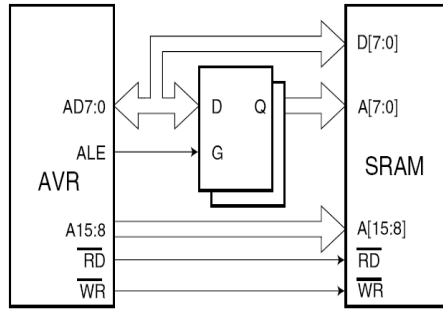


그림 22 AVR에 연결된 외부 SRAM

외부 메모리 인터페이스를 위한 제어 비트들은 MCU 제어 레지스터인 MCUCR, 외부 메모리 제어 레지스터 A, XMCRA, 그리고 외부 메모리 제어 레지스터 B, XMCRB 에 위치해 있습니다. 아래 그림은 외부 데이터 메모리를 어떻게 제어하는 지를 설명하고 있습니다. T1 ~ T4까지 상태에서의 동작에 대해 잠시 설명 하겠습니다. T2에서는 System Clock이 Low to High가 되는 순간 A15~A0까지의 전 어드레스가 출력되며, 하위 어드레스 출력핀이 데이터버스와 함께 핀을 공유하여 사용하기 때문에, T2에서 ALE가 High에서 Low로 자동 변경되는 데, 이 신호를 74xx573의 Latch Enable에 입력하여 주면, 하위 어드레스인 A7~A0가 값이 유지되고, T2에서 하위어드레스가 나오던 핀에서 T3가 되면 데이터 버스로 변경되어 값이 D7~ D0 값이 출력될 때, Read나 Write 신호 발생을 받아서, 램과 같은 패러럴 형태의 디바이스에 연결하여 주면, 그 디바이스에 A15~A0, D7~D0의 신호를 읽거나 쓸 수 있게 됩니다.

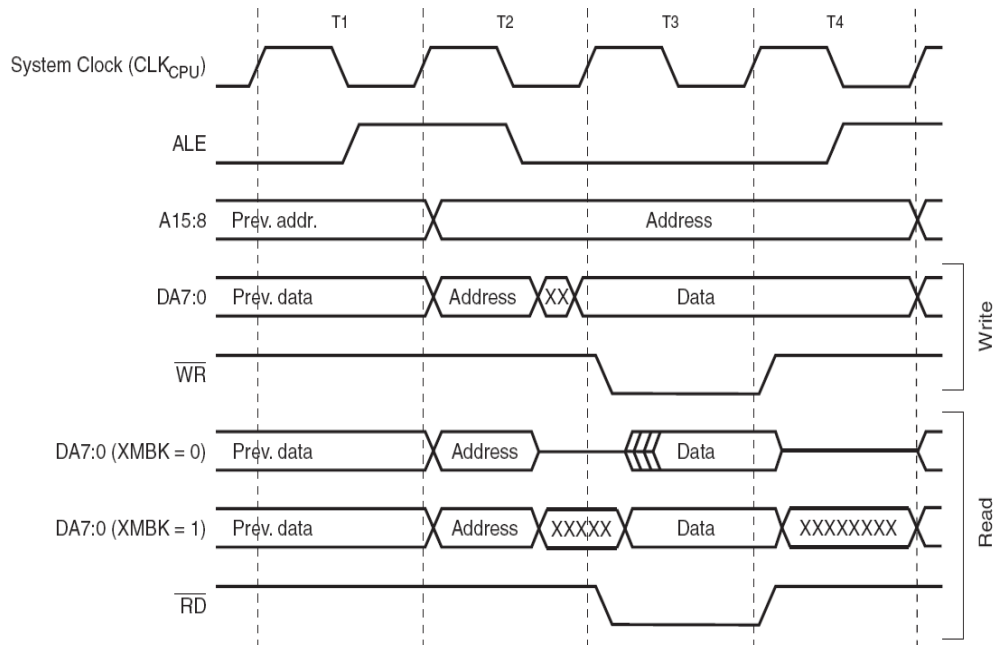


그림 23 ATMEGA128에서 패러렐로 인터페이스 된 외부 램의 동작 타이밍 도

즉, 이 디바이스의 위치에 램을 붙이지 않고, 패러럴 형태의 I/O를 붙여 놓게 되면, 그 값이 디바이스에서 사용이 가능해지며, 주소 값을 가진 I/O가 되며, 데이터를 입출력 할 수 있게 됩니다. 위에서 설명한 대로, ALE가 High가 되면 74573이 동작되면서 D에서 Q로 T2에 Address가 옮겨지

게 된다. 즉, T2에서 AVR에서의 하위어드레스가 573으로 옮겨지며, T3에서는 ALE가 Low이므로 잠기면서 573에 그 하위 어드레스 값이 유지되게 됩니다. T4에서 ALE가 다시 High가 되면서 573에 저장되었던 AVR 하위 어드레스 값과 A15:8의 상위 어드레스 값과 DA7:0의 데이터 값이 그 어드레스 램에 Write되거나 그 어드레스 램의 데이터 값이 AVR의 데이터 라인에 READ해서 실리게 됩니다.

### 3.11 KT-M128 키트에서의 Memory Mapped I/O의 회로 구성

ATMEGA128에서는 상위 어드레스(A15~A8)가 PC7~PC0 핀에 물려 있으며, 하위 어드레스 (A7~A0)가 PA7~PA0에 물려 있으며, 8비트 데이터 버스도 PA7~PA0에 물려 있습니다.

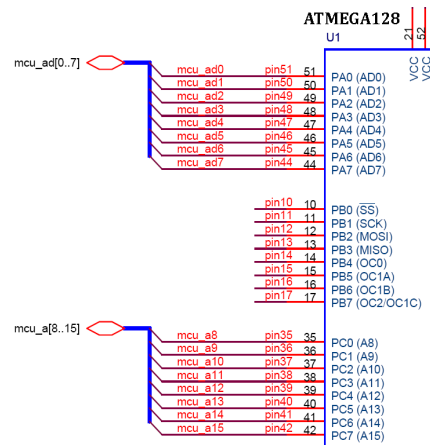


그림 24 ATMEGA128에서의 어드레스/데이터버스

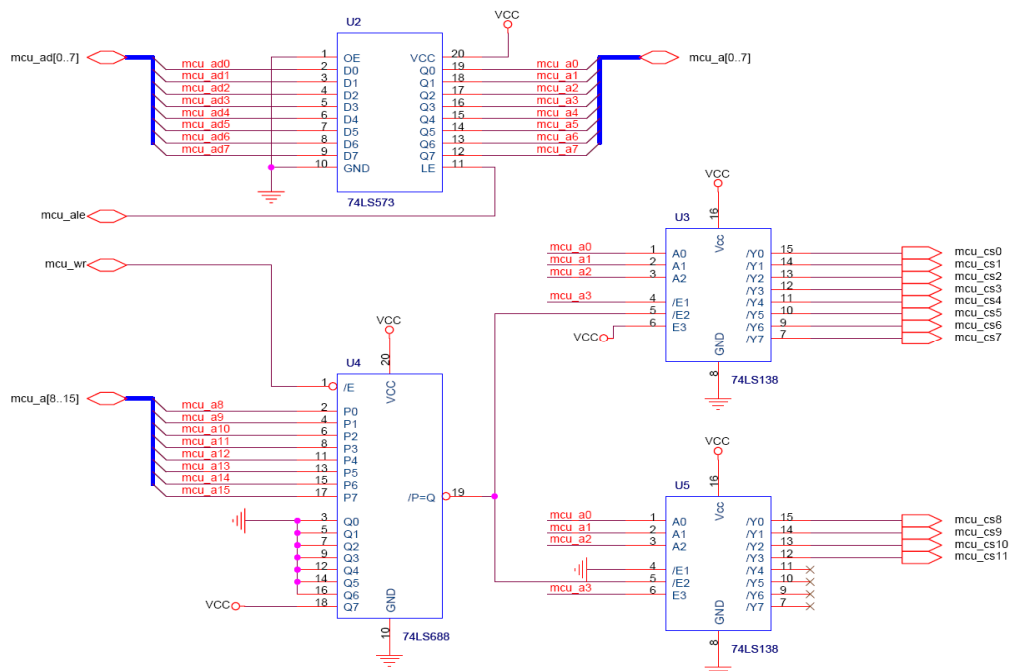


그림 25 KT-M128 키트의 메모리 주소 관련 회로도

메모리 주소값은 74xx688에서 Write시에만 A8~A14 상위 어드레스 값이 Low이고, A15가 High인 경우 19번핀에서 Low가 출력되게 되며, 아니면 High가 출력됩니다. 즉, 0x80XX 번지에서는 74xx138인 U3와 U5가 선택되며, 이 때에만 본 키트에 붙어 있는 Memory Mapped I/O의 동작이 가능합니다. 또한, A3의 값이 따라서, Low일 경우는 U3가, High일 경우는 U5가 선택됩니다. 또한, A2~A0 값에 의하여 나머지 칩의 CS(Chip Select)핀이 선택되어 Low가 출력되는 데, 아래 표와 같이 됩니다.

| A3 | A2~A0 | Low가 출력되는 CS핀 |
|----|-------|---------------|
| 0  | 000   | mcu_cs0핀      |
| 0  | 001   | mcu_cs1핀      |
| 0  | 010   | mcu_cs2핀      |
| 0  | 011   | mcu_cs3핀      |
| 0  | 100   | mcu_cs4핀      |
| 0  | 101   | mcu_cs5핀      |
| 0  | 110   | mcu_cs6핀      |
| 0  | 111   | mcu_cs7핀      |
| 1  | 000   | mcu_cs8핀      |
| 1  | 001   | mcu_cs9핀      |
| 1  | 010   | mcu_cs10핀     |
| 1  | 011   | mcu_cs11핀     |

표 3 A3~A0에 의한 I/O되는 CS핀의 주소결정

즉, 0x80X0 ~ 0x80XB 번지까지 주소 값에 의해 CS0~CS11이 선택되는 데, 기억상의 편의를 위하여, 예제 소스 상에는 0x8000~0x800B번지로 정하였습니다.

각, CS0~CS11 는 74xx574 칩으로 입력되어, 그 순간 전 데이터 값이 들어오게 되는 데, 이 이후 바로 CS값이 Low에서 High로 바뀌면서 Dx 값이 Qx값으로 전달되면서 I/O 회로로 데이터값이 전달됩니다. 이 때, CS 값이 Low에서 High로 바뀌는 이유는 PC(Program Counter)에 의하여 클럭 변경 시 어드레스가 다음 프로그램 값에 따라서 변경되기 때문입니다.

### 3.12 LED 부분 회로도와 소스코드의 작성

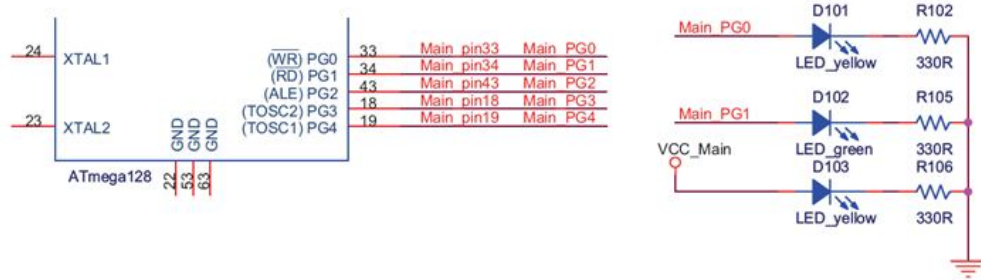


그림 26 KD-128PRO-C의 메인보드인 AB-M128PRO-A 보드의 LED부분 회로도

\* 소스코드

```
DDRG = 0x03; // 0b00000011
while(1){
    PORTG = 0x01; // 0b00000001
    delay(500);
    PORTG = 0x02; // 0b00000010
    delay(500);
}
```

### 3.13 스위치 부분 회로도와 소스코드의 작성

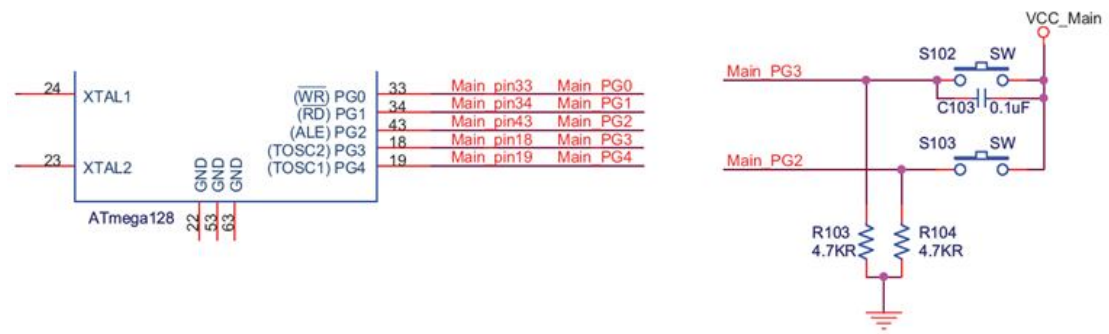


그림 27 KD-128PRO-C의 메인보드인 AB-M128PRO-A 보드의 스위치부분 회로도

\* 소스코드

```
DDRG = 0x03; // 0b00000011
while(1){
    if((PING & 0x04)){
        PORTG = 0x01;
    }
    if((PING & 0x08)){
        PORTG = 0x02;
    }
}
```

### 3.14 세븐 세그먼트 부분 회로도와 소스코드의 작성

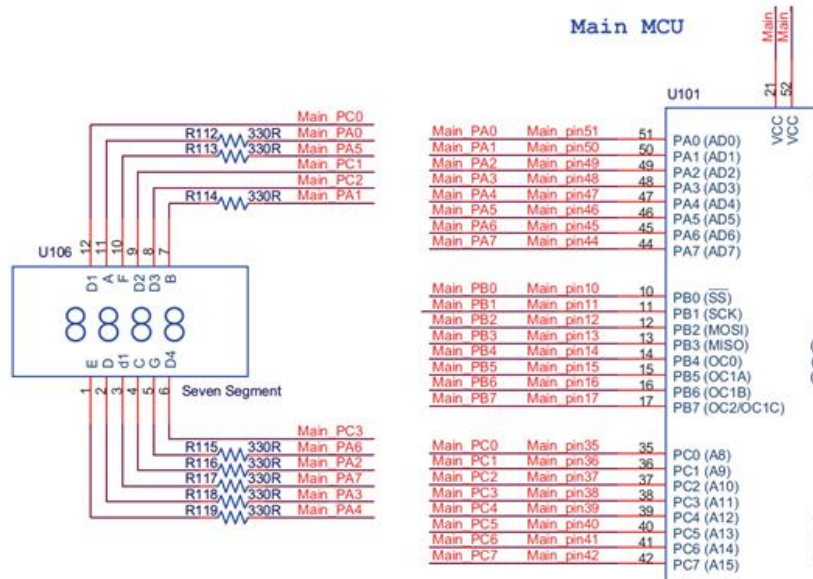


그림 28 KD-128PRO-C의 메인보드인 AB-M128PRO-A 보드의 세븐세그먼트부분 회로도

#### \* 소스코드

설명 : Jari[0], Jari[1], Jari[2], Jari[3]은 세븐세그먼트의 자리를 말한다.

예를 들어서 Jari[3] = 8을 넣으면, 3번째 세그먼트에 8이라는 값이 출력된다.

소스 :

```
char number[10] = {0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x67};
while (1){
    PORTC = ~0x01;
    PORTA = number[jari[3]];
    delay(10);
    PORTC = ~0x02;
    PORTA = number[jari[2]];
    delay(10);
    PORTC = ~0x04;
    PORTA = number[jari[1]];
    delay(10);
    PORTC = ~0x08;
    PORTA = number[jari[0]];
    delay(10);
}
```