Sparkfun Tutorial: https://www.sparkfun.com/tutorials/category/1

C Compiler

The C compiler that we will use is the WinAVR GCC compiler. It is a free and very useful c-compiler designed for the ATMEL AVR 8-bit compilers. This compiler is available separately but it is included in the development environment we will be using Atmel Studio 7.0 (also free). The development environment brings together in one program the ability to create a C program (editor), compile or build the program and download to the microcontroller.

A recent development is that the company Microchip purchased Atmel so the former "Atmel Studio" is now called "Microchip Studio for AVR® and SAM Devices". The only thing that has changed is the name it is still Atmel Studio.

Microchip Studio can be downloaded at:
https://www.microchip.com/en-us/development-tools-tools-and-software/microchip-studio-for-avr-and-sam-devices

During the install at one step you will be asked to select the Architecture. We only need AVR not UC3 or SAM but it doesn't hurt to select these also. The program will take a bit longer to load if you choose to include these extra elements. The rest of the installation is pretty straight forward. Make sure you do the complete install and don't choose to leave anything out because Atmel Studio will not install correctly if you leave out any of the components.

Once Atmel Studio 7.0 is installed.

- Select **New Project**
- Select **GCC C Executable Project**
    - At the bottom of the screen Specify a name for the project and a location to save it
    - Check the **Create Directory for solution** box at the bottom right of the screen
    - Click **OK**
- The Device Selection screen appears next where the microcontroller is selected. Our chip is the Atmel atmega88PA.
    - Select **atmega88PA**
    - Click **OK**

A C program template now appears.

Now you can create a C program or if you are doing the Sparkfun tutorial simply copy the **blink_1MHz.c** program into the C program window of your project. Note delete or copy over the existing C code in the template.

Now the C program needs to be complied and linked. This is done with the Build command.

- Select the **Build menu** and then **Build Solution**

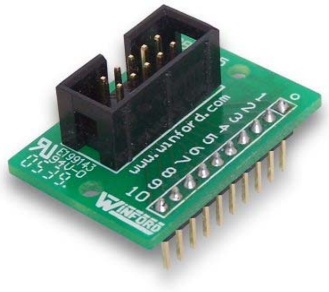
This compiles the C program and links any necessary libraries and creates a **.hex** file that can be loaded onto the microcontroller. The **.hex** file is located in a directory called **debug** that is in your project directory.

Programming the Microcontroller

After a C program has been written and successfully compiled and linked the result is a .hex file that needs to be downloaded to the flash memory of the microcontroller. The microcontroller then runs the program. There are several different ways the program can be downloaded to the microcontroller. We will use the In System Programming (ISP) method. To use this method, the appropriate connections between the programmer that is connected to a PC either through a USB of serial connection and your microcontroller on your breadboard needs to be made. The ATMEL document below is a good resource for a number of hardware issues. Refer to section 4 Connecting ISP lines for the proper connections.

AVR042: AVR Hardware Design Considerations
http://www.atmel.com/dyn/resources/prod_documents/doc2521.pdf

The standard ISP connector is either a 6 pin or 10 pin two row connector. This connector is not easily interfaced with a breadboard. A convenient converter is available that converts the two row connector to a single row connector that can be easily plugged into a breadboard.

VTG is the 5 volt supply, GND is ground, MOSI, SCK and MISO are the spi bus connections on the microcontroller and RST is the reset pin on the microcontroller. Check the data sheet for the microcontroller to determine which pins on the microcontroller correspond to these pins.

The programmer that we are using is the ATMEL AVRISP2 programmer. It connects to the computer using a USB interface and connects to the microprocessor with a 6 pin two row connector. An adapter is attached that converts the two row connector to a single row connector that is easily interfaced with your breadboard. **Do not poke wires into the connector of the programmer as is described in the Tutorial as this will damage the connector.** There are several of these programmers in the Controls Lab on the table by the resistors. They can be used on any of the computers in the Controls Lab. When you are done with the programmer please return it to the table by the resistors.

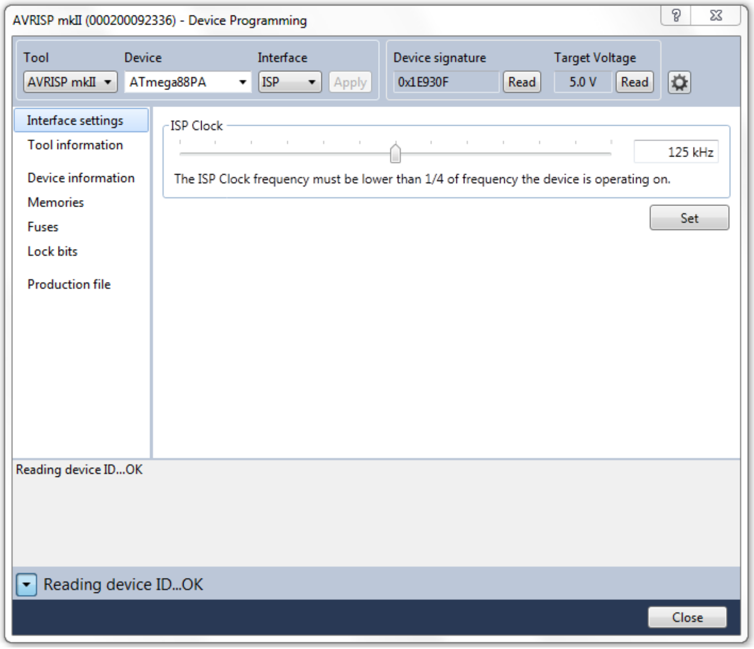
After the programmer is connected power up your microcontroller. If you are prompted to update the firmware on the programmer do so. The light on the front of the programmer should now be green. If it is not green go to the **Help menu** in AVR Studio and select **AVR Tools**, from the drop down menu select **AVRISP mkII User Guide** and then select **Troubleshooting**. This will describe the most common problems. The most common problem is that you do not have the connection from the programmer to the programmer correct or another connection on the microcontroller is incorrect. Also sometimes it is necessary to unplug the programmer and plug it back in to reset it.

Once the ISP connections are made the .hex file can be downloaded to the microcontroller.

- Run **Atmel Studio 7.0**
- Select your project that you compiled earlier.
- From the **tools menu** at the top of the screen select **Device Programming**. Note that this is on the toolbar also.
  - **Tools > Device Programming**
- From the **tool** drop down menu in the Device Programming Window Select **AVRISP mkII**. Note the programmer must be connected and your microcontroller powered to see this selection.
  - **Tool** > **AVRISP mkII**
- From the **device** drop down menu in the Device Programming Window Select **atmega88PA**.
  - **Device** > **atmega88PA**

  bar at the top of the screen select **Connect to Standard AVR Program** (This is a small box with AVR as the label). Select **AVRISP mk II** and click **Connect**.

- Click the **Apply** button. The Device Programming Window should now appear as the window below.
- Under Device Signature select **Read**. This reads the device signature and compares it to the device you have specified. The Device Programming Window should now appear as the window below.


AVRISP mkII (000200092336) - Device Programming

Tool: AVRISP mkII
Device: ATmega88PA
Interface: ISP
Device signature: 0x1E930F
Target Voltage: 5.0 V

ISP Clock: 125 kHz

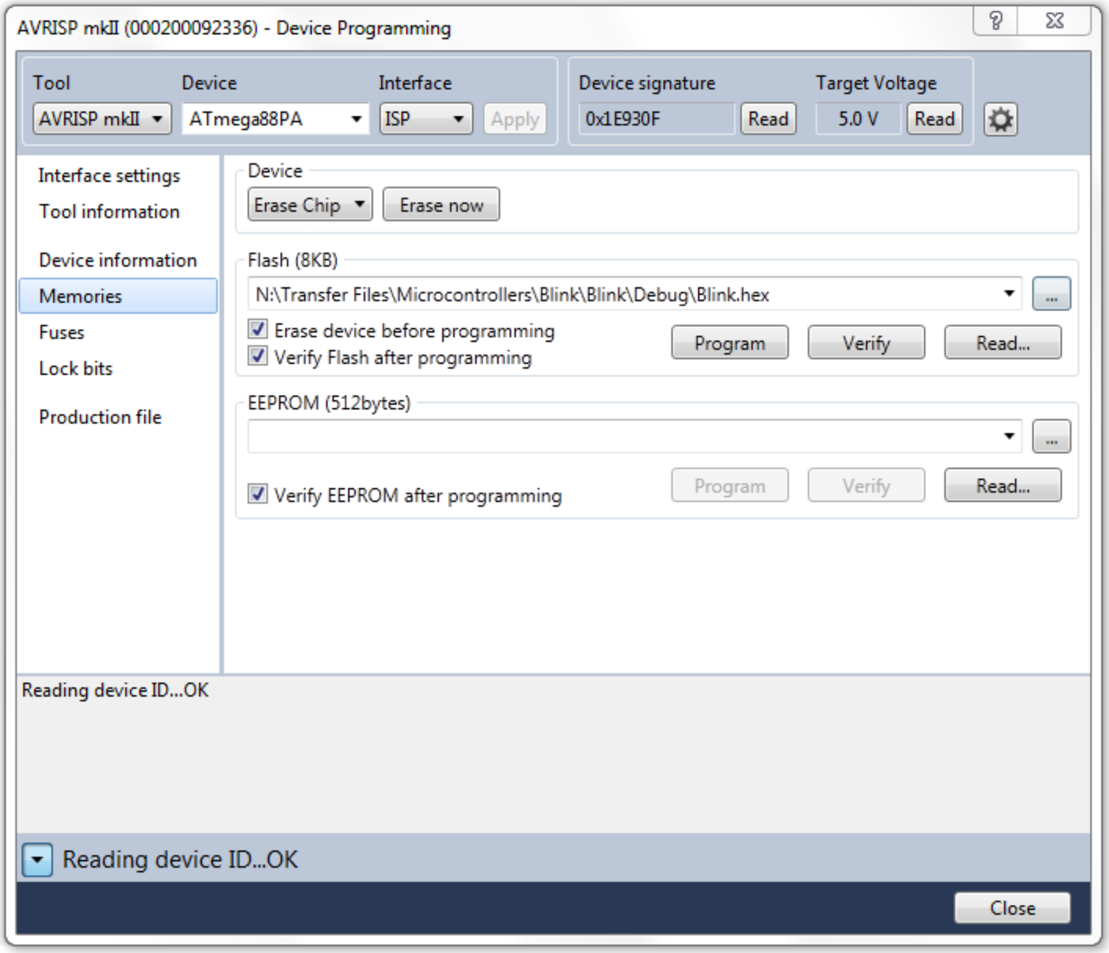
The ISP Clock frequency must be lower than 1/4 of frequency the device is operating on.

Reading device ID...OK

Reading device ID...OK

- The **Read Signature** will check connectivity. If the signature is not read correctly nothing else will work so if this is the case determine what the problem is. The most likely problem is that the connections between the programmer and the microcontroller are not correct. Note a window at the bottom of the programming window shows the status of each command.

- Once the signature is read correctly select **Memories** from the menu on the left of the Device Programming Window. The Device Programming Window should now appear as the window below except for the directory where your project is stored. This should be the location of your project. Note that the file to be programed is the .hex file not the .elf file that appears by default. The .elf file will also work but it contains extra code for debugging that is not necessary.



AVRISP mkII (000200092336) - Device Programming

Tool: AVRISP mkII

Device: ATmega88PA

Interface: ISP

Device signature: 0x1E930F

Target Voltage: 5.0 V

Device: Erase Chip

Flash (8KB): N:\Transfer Files\Microcontrollers\Blink\Blink\Debug\Blink.hex

☑ Erase device before programming

☑ Verify Flash after programming

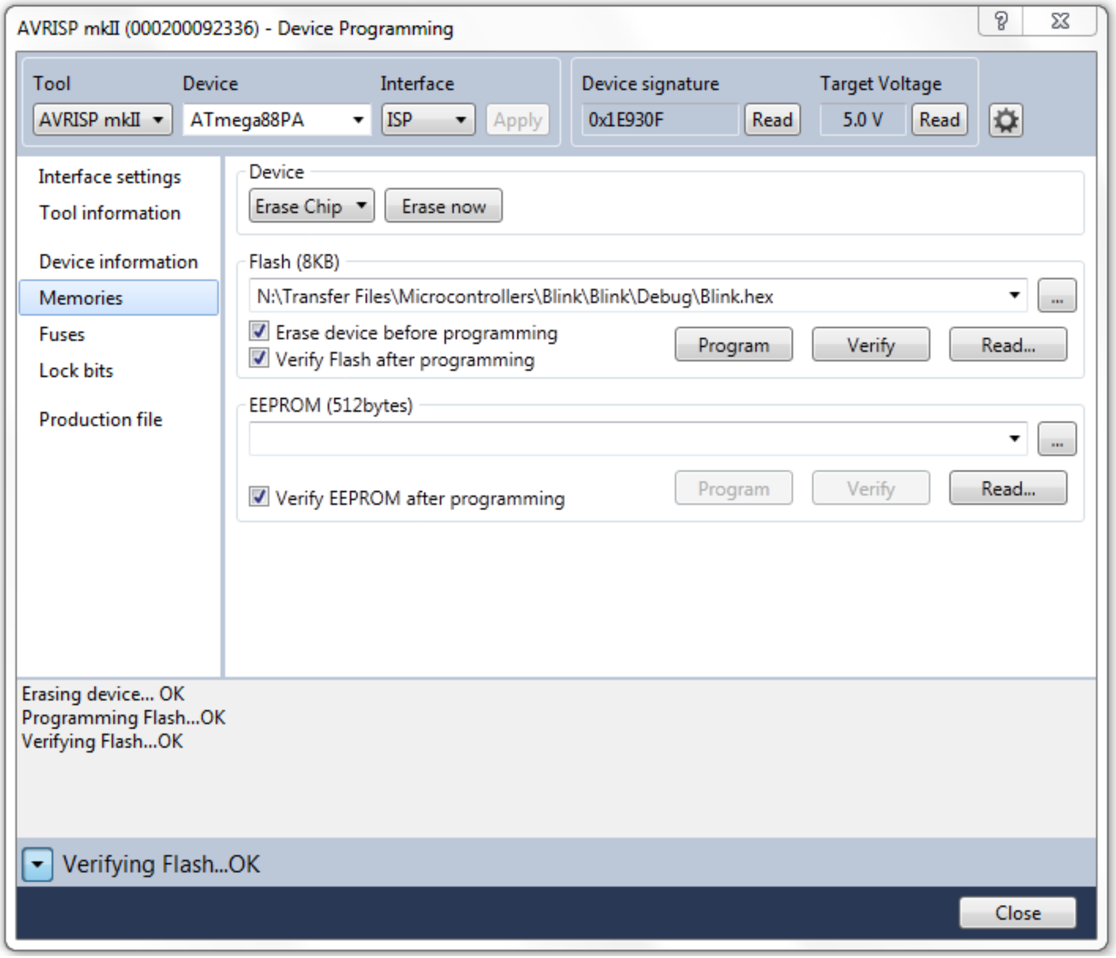
EEPROM (512bytes)

☑ Verify EEPROM after programming

Reading device ID...OK

Reading device ID...OK

- After verifying the location of the .hex file Select **Program.** The Device Programming Window should now appear as the window below.



AVRISP mkII (000200092336) - Device Programming

Tool: AVRISP mkII
Device: ATmega88PA
Interface: ISP
Device signature: 0x1E930F
Target Voltage: 5.0 V

Device: Erase Chip

Flash (8KB): N:\Transfer Files\Microcontrollers\Blink\Blink\Debug\Blink.hex

☑ Erase device before programming
☑ Verify Flash after programming

EEPROM (512bytes)

☑ Verify EEPROM after programming

```
Erasing device... OK
Programming Flash...OK
Verifying Flash...OK
```

Verifying Flash...OK

- The program is now download to the microcontroller and will begin running.

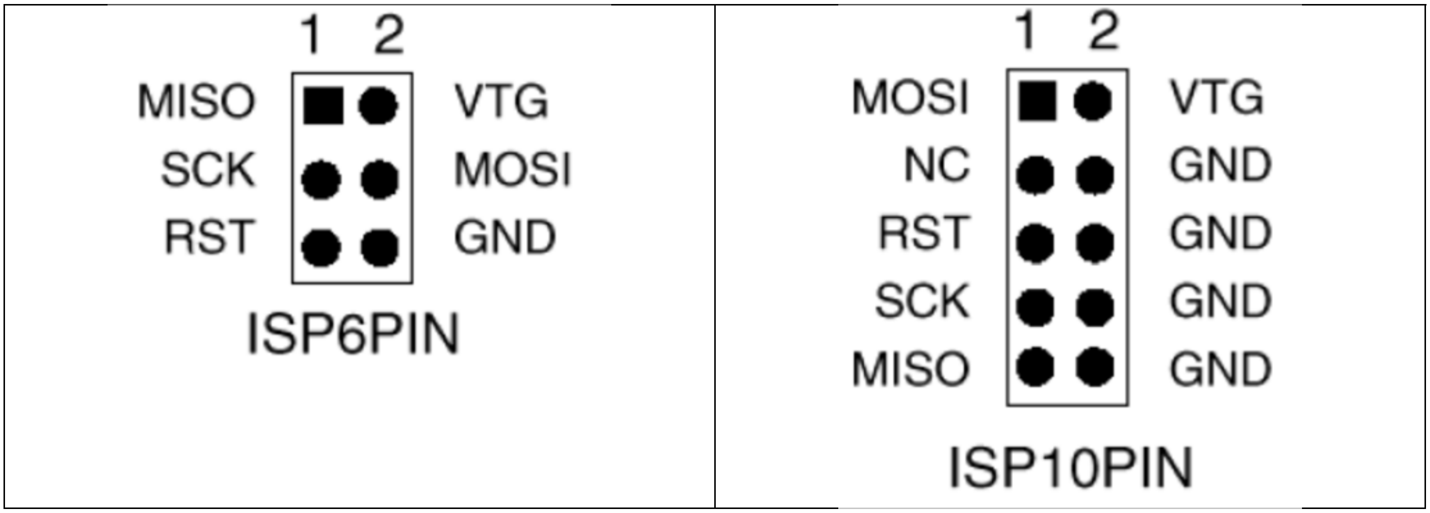
Appendix A: **AVR042: AVR Hardware Design Considerations (section 4)**
http://www.atmel.com/dyn/resources/prod_documents/doc2521.pdf

4 Connecting ISP lines

The In-System Programmable (ISP) lines are used for programming the Flash, EEPROM, Lock-bits and most Fuse-bits in all AVRs (except the ATtiny11 and ATtiny28). This feature makes it possible to program the AVR on the last stage of production of a target application board, reprogram if SW bugs are identified late in the process, or even update the AVR in the field if required. It is therefore highly recommended to always design the target application board so that the ISP connectors can be accessed in some way, - just in case.

Two standard connectors are provided by the Atmel ISP programmers; A 6 pin and a 10 pin connector. These are seen in Figure 4-1. In addition to the data lines (MOSI and MISO) and the bus clock (SCK), target voltage VTG, GND and RESET (RST) are provided through these connectors.

Figure 4-1. Standard ISP connectors on STK500, AVR ISP and STK200/STK300



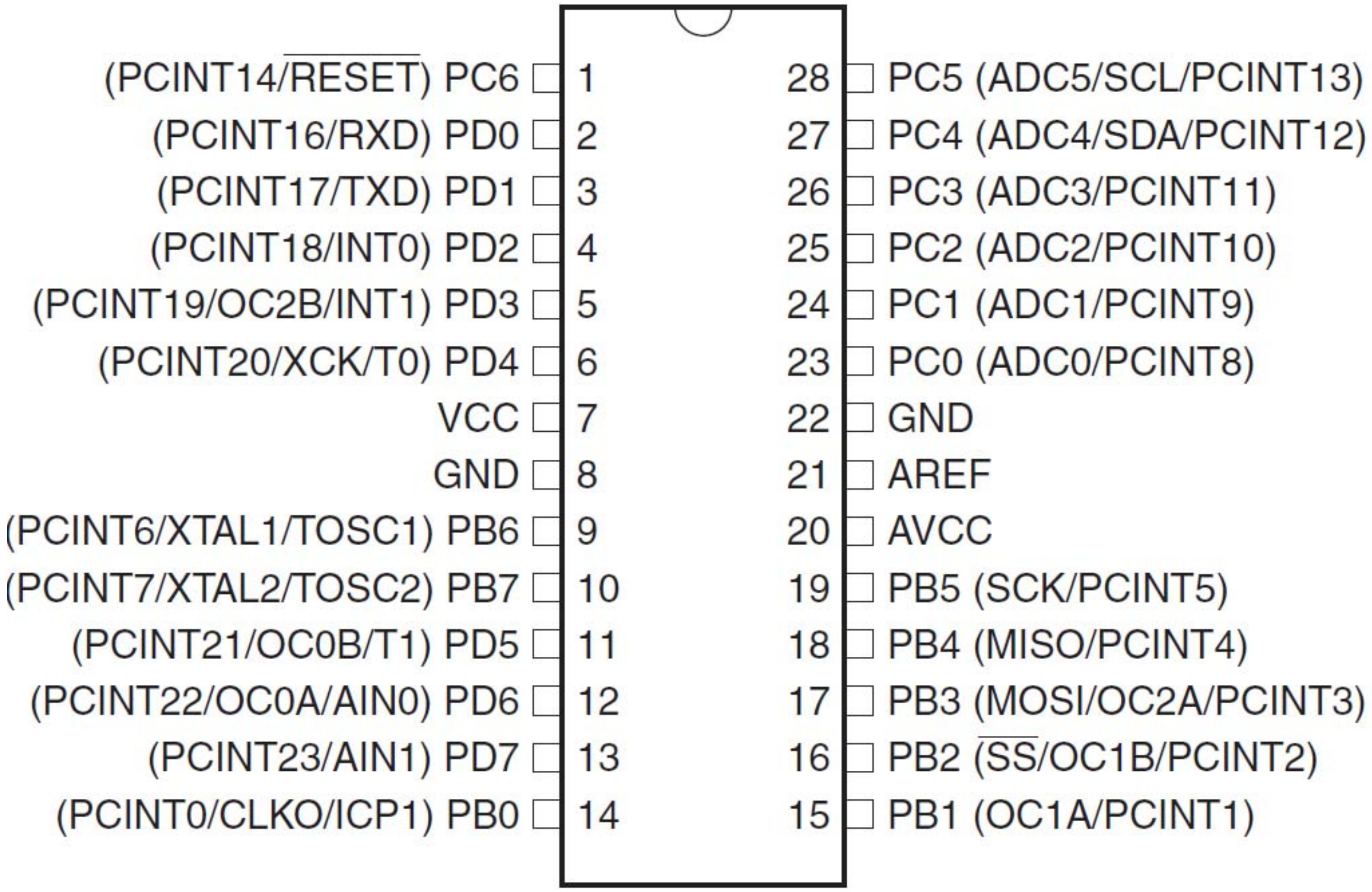
Some ISP programmers are supplied by the target power supply. In this way they easily adapt to the correct voltage level of the target board. Other ISP programmers, like STK500, can alternatively power the target board via the VTG line. In that case it is important that the power supply on the target is not switched on. Read the User Guide of the ISP programmer to find out what capabilities your programmer has and what kind of physical interface it has.

The ISP lines are on most devices located at the same pins as the Peripheral Serial Interface (SPI), or else on pins that can be used for other purposes. Consult the device data sheet to find out which lines are used for ISP. In case other devices than the AVR is connected to the ISP lines the programmer must be protected from any device, other than the AVR, that may try to drive the lines. This is especially important with the SPI bus, as it is similar to the ISP interface. Applying series resistors on the SPI lines, as depicted in Figure 4-2, is the easiest way of doing this. The AVR will never drive the SPI lines in a programming situation, since the AVR is held in RESET to enter programming mode – and RESET'ing the AVR tri-states all pins.

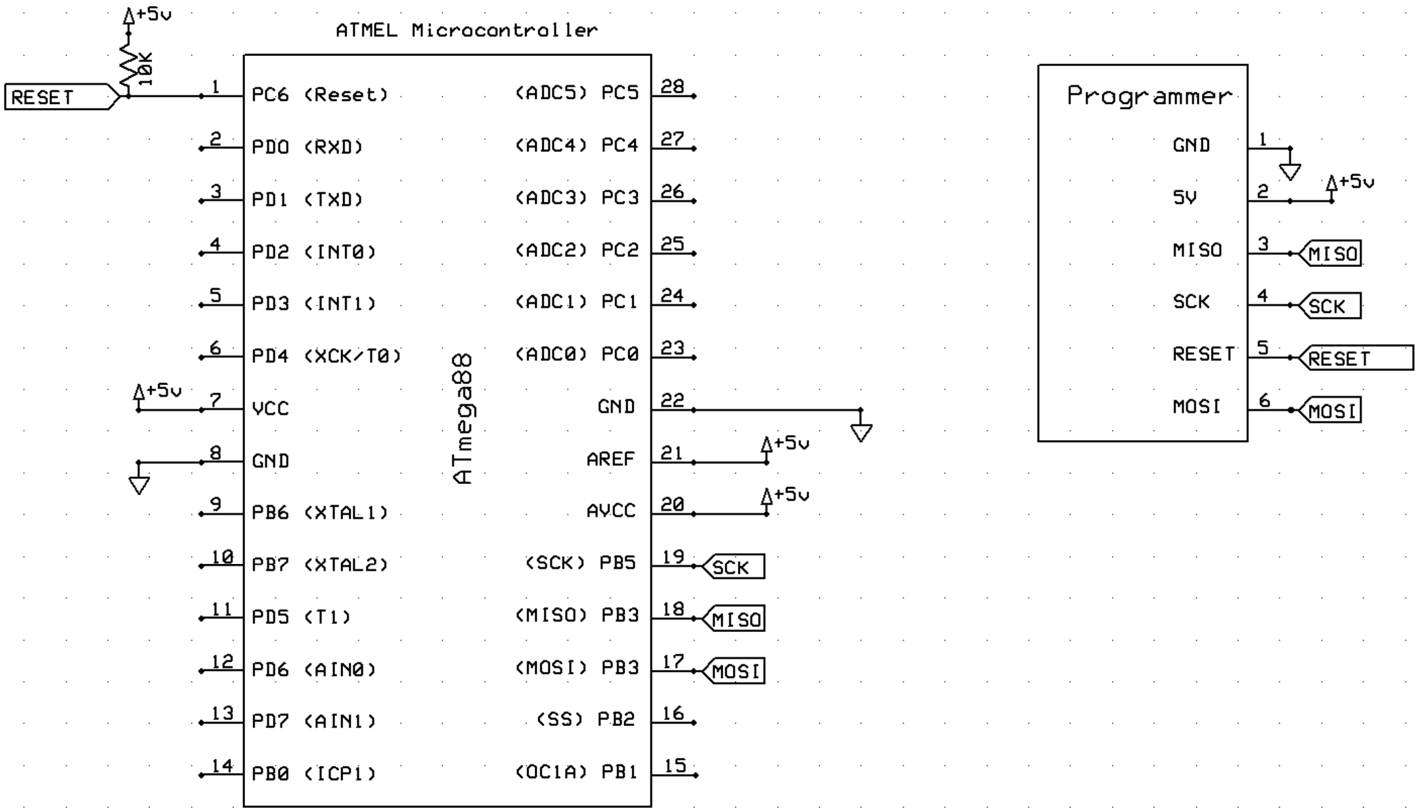
Appendix B: ATmega88PA_manual – Pinouts

http://www.atmel.com/Images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Summary.pdf

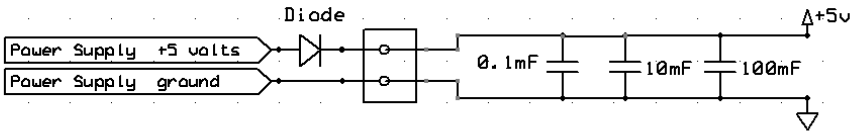
http://www.atmel.com/Images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet.pdf



Appendix C: Basic Microcontroller Setup Schematic



Power Supply Diode and Decoupling Capacitors


Diode
+5v
Power Supply +5 volts
Power Supply ground
0.1mF
10mF
100mF