APPENDIX

# Deciphering the ATMega Datasheet and Arduino Schematics

At the heart of all Arduinos is an Atmel microcontroller. This appendix does not summarize the features of all the microcontrollers in all the Arduinos, but it is a useful exercise to investigate an ATMega datasheet to get a better idea about how it works. Further, taking a look at the open source schematics for the Arduino Uno will make it easier to understand how an Arduino actually works.

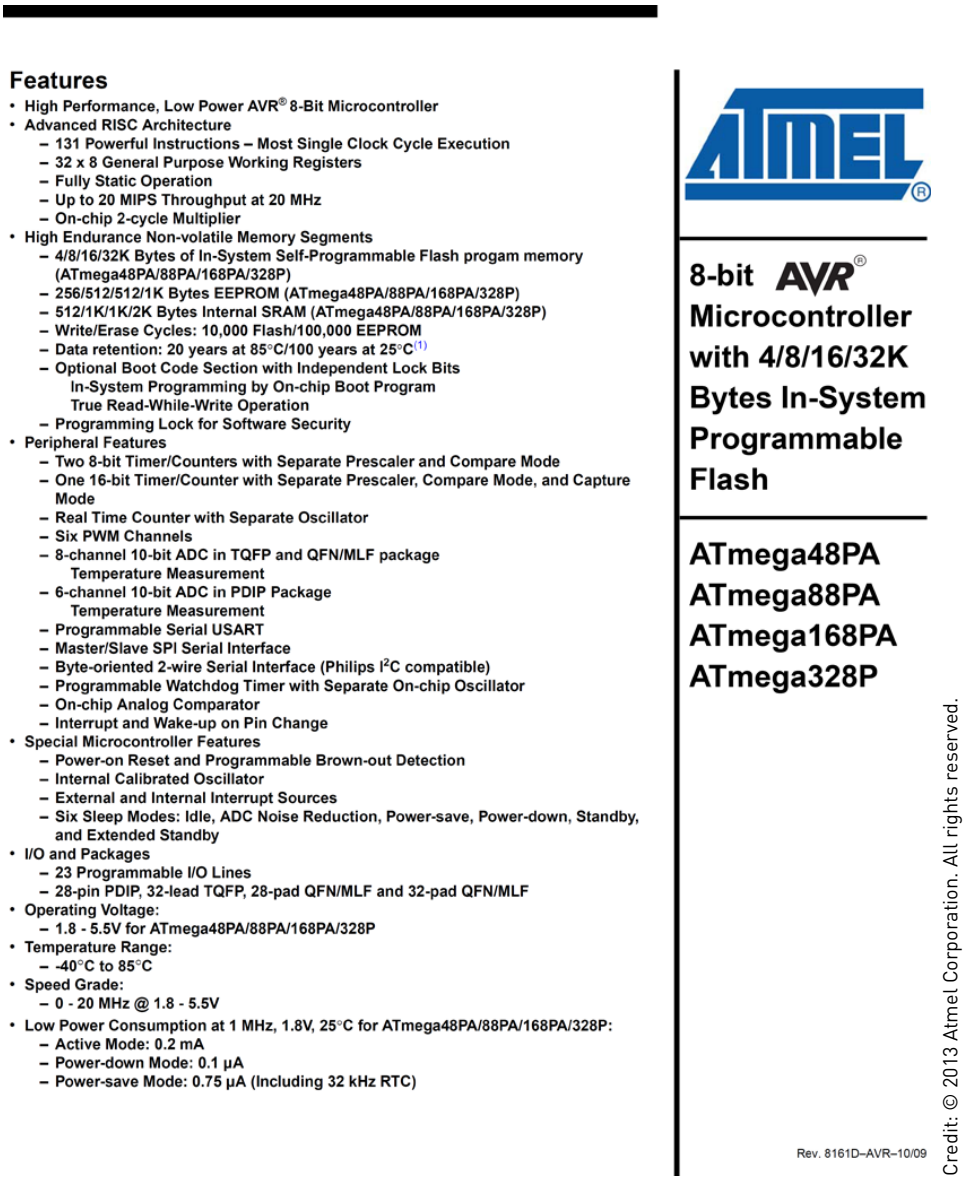
## Reading Datasheets

One of the most important skills that you can develop as an engineer is the ability to read datasheets. Just about any electronic component that you can buy has an associated datasheet that contains info about the technical limits of the part, instructions on how to use its capabilities, and so forth.

### Breaking Down a Datasheet

Consider the datasheet for the Atmel ATMega 328p, for instance. Recall that the ATMega 328p is the microcontroller unit (MCU) used in the Arduino Uno and many Arduino clones. Finding a datasheet can often be the trickiest part. I recommend just doing a Google search for "ATMega 328p datasheet" and looking for the first PDF link from Atmel. The datasheets for the MCUs used in the Arduinos can also be found on the hardware page for each board on the

www.Arduino.cc website. When you have the datasheet in hand, start by reviewing the first page (see Figure A-1). In most cases, the first page tells you all you need to know about the features of that MCU.



## Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32K Bytes of In-System Self-Programmable Flash progam memory (ATmega48PA/88PA/168PA/328P)
  - 256/512/512/1K Bytes EEPROM (ATmega48PA/88PA/168PA/328P)
  - 512/1K/1K/2K Bytes Internal SRAM (ATmega48PA/88PA/168PA/328P)
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    - Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I²C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
  - 1.8 - 5.5V for ATmega48PA/88PA/168PA/328P
- Temperature Range:
  - -40°C to 85°C
- Speed Grade:
  - 0 - 20 MHz @ 1.8 - 5.5V
- Low Power Consumption at 1 MHz, 1.8V, 25°C for ATmega48PA/88PA/168PA/328P:
  - Active Mode: 0.2 mA
  - Power-down Mode: 0.1 µA
  - Power-save Mode: 0.75 µA (Including 32 kHz RTC)

8-bit AVR® Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash

ATmega48PA
ATmega88PA
ATmega168PA
ATmega328P

Rev. 8161D–AVR–10/09



**Figure A-1:** The first page of the ATMega 328p datasheet

From a quick glance at the datasheet, you can learn a considerable amount about the microcontroller. You can ascertain that it has 32KB of programmable flash memory, that it can be reprogrammed about 10,000 times, and that it can operate from 1.8V to 5.5V (5V in the case of the Arduino). You can also learn how many inputs/outputs (I/Os) it has, what special functions it has built in (like hardware serial peripheral interface [SPI] and $I^2C$ interfaces), and what resolution its analog-to-digital converter (ADC) is.

**NOTE** This datasheet is actually hundreds of pages, and there could probably be an entire book dedicated just to interpreting it, so I won't go much further here. However, throughout the remainder of this appendix, I do point out several more important topics to look out for.

Datasheets as long as this one generally have PDF bookmarks built in that make it easier to find what you're looking for. Of particular interest for your Arduino adventures may be information about I/O ports, the timers, and the various hardware serial interfaces. As one more example, consider Figure 13-1 from the datasheet's I/O section in the PDF, which is shown here as Figure A-2 for your convenience.

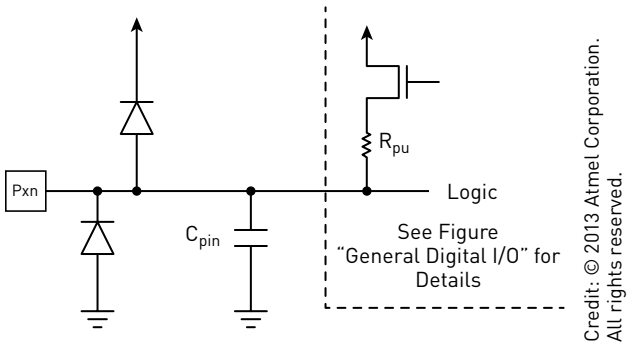



**Figure A-2:** I/O pins diagram

Diagrams like this one can be found throughout the datasheet, and can give you a deeper insight into how your Arduino is actually working. In this example, you can see that the I/O pins all have protection diodes to protect them from excessively high or negative voltages. It's also important to observe that there is a known pin capacitance, which could have significant implications when trying to determine the rise and fall times when switching the value of a pin.

## Understanding Component Pin-outs

All datasheets will include the pin-out for the device in question, which clearly illustrates the functions of each pin. Particularly for microcontrollers, pins may have multiple functions, so understanding the pin-out can be critical for grasping what each pin can and cannot do. Consider the pin-out of the ATMega 328p (see Figure A-3). Understanding the pin-out of the microcontroller at its heart will make it easier to understand the Arduino Uno schematic, which you'll look at in the next section.

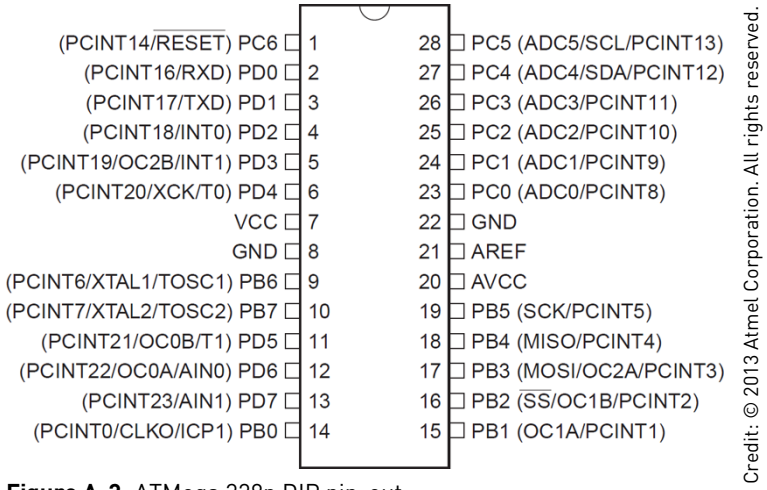



**Figure A-3:** ATMega 328p DIP pin-out

Note that the pin-out indicates how you can find the pin number on the actual chip. The half circle at the top of the pin-out corresponds to a similar half circle on the actual integrated circuit. Look at the chip in your Arduino and you'll see this half circle; now you know that the pin immediately to its left is pin 1.

You'll also probably notice some abbreviations that you may not be familiar with. They are defined here:

- VCC refers to voltage supply to the chip. In the case of the Arduino, VCC is 5V.
- AVCC is a separate supply voltage for the ADC. For the Arduino, it is also 5V.
- AREF is broken out to a pin. So, you can choose an arbitrary voltage below 5V to act as the reference for the ADC if you desire.
- GND is, of course, the ground connection.

The rest of the pins are all general-purpose I/O. Each is mapped to a unique pin number in the Arduino software so that you don't have to worry about the port letter and number.

The labels in parentheses represent alternative functions for each pin. For example, pins PD0 and PD1 are also the Universal Synchronous/Asynchronous Receiver/Transmitter (USART) Receive (RX) and Transmit (TX) pins, respectively. Pins PB6 and PB7 are the crystal connection pins (XTAL). In the case of the Arduino Uno, an external 16 MHz ceramic resonator is connected to these pins, so you cannot use these for general-purpose I/O. If you have trouble deciphering the pin labels, you can usually learn more about what they mean by searching the rest of the datasheet for those terms. The Arduino website has a diagram illustrating how the ATMega pins are connected to numbered pins on the Arduino board. You can find it at `http://arduino.cc/en/Hacking/PinMapping168`, and it is shown in Figure A-4.

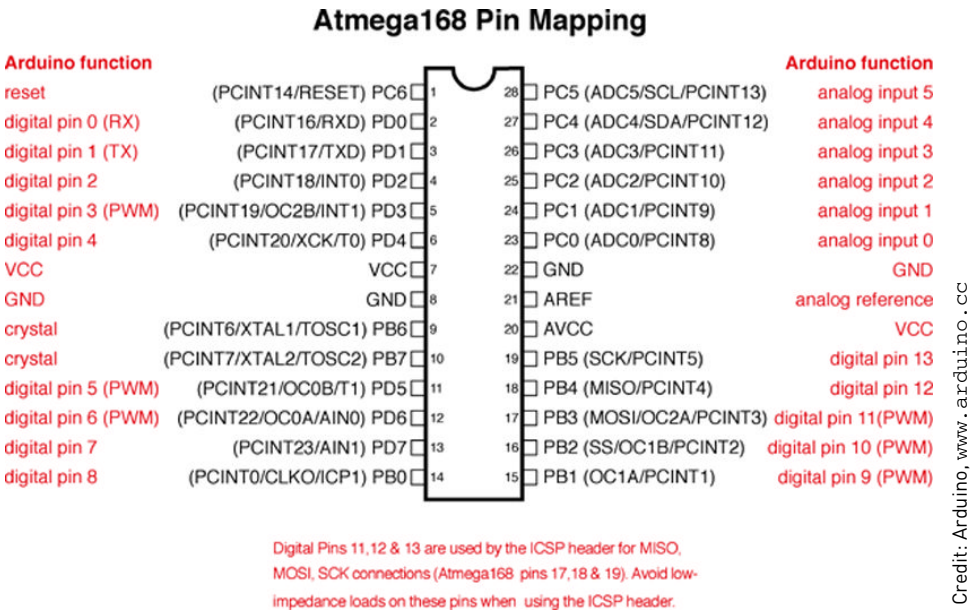



**Figure A-4:** Arduino ATMega Pin Mapping

## Understanding the Arduino Schematic

Perhaps one of the best ways to learn about electrical design is to analyze the schematics of existing products, such as the Arduino. Figure A-4 shows the schematic for the Arduino Uno.

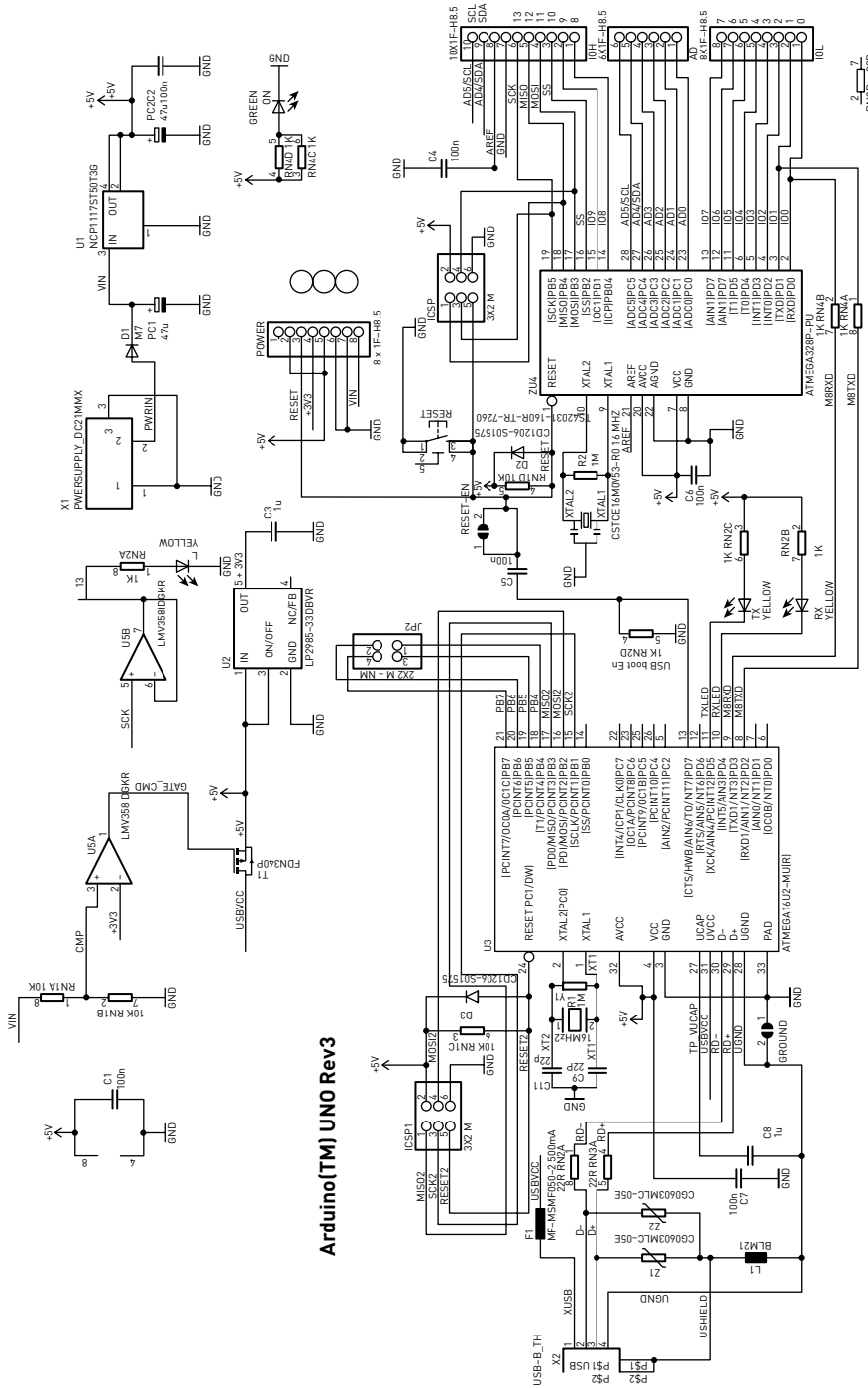



**Figure A-5:** Arduino Uno Rev 3 schematic

Can you match all the parts to the parts that you can see on your Arduino Uno? Start with the main MCU (Part ZU4 in the schematic), the ATMega328p, and all the breakout pins. Here, you can easily identify which ATMega ports/pins map to the pins that are available to you in the integrated development environment (IDE). Earlier in this appendix, you observed that PD0 and PD1 were connected to the USART TX and RX pins. In the Arduino schematic, you can indeed confirm that these pins connect to the corresponding pins on the 16U2 (8U2 on revisions 1 and 2) USB-to-Serial converter chip. You also know that there is an LED connected (through a resistor) to pin 13 of the Arduino. In the schematic, you can see that pin 13 is connected to pin PB5 on the ATMega. But where is the LED? By using net names, you can indicate an electrical connection between two points on a schematic without actually drawing all the lines. Having every wire shown in a schematic might get confusing very quickly. In the case of PB5, you can see that the wire coming out of the MCU is labeled *SCK*, and that there is a similarly labeled wire at the top of the schematic feeding through a buffer into a resistor and the familiar debug LED.

Most schematics that you'll find are done in a style similar to this one, with lots of labeled nets that connect without direct wires. Continue to analyze the Arduino schematic until you understand where all the signals are going. See how many components you can match to the actual board.