All Things Digital

Amateur Radio for the 21st Century

005

Robert C. Mazur, VA3ROM
E: va3rom@gmail.com
W: www.va3rom.com

Article first published in the November-December 2012 issue of <u>The Canadian Amateur</u>

Portions of this material appeared in APRS Thunder Bay, Vol. 3, No. 22 (2012), "PICAXE Power for Hams!" and "PICAXE Apps: Radio or Not!"

MCU + BASIC & U = FUN!

Okay, here we go into something probably new to many Hams—the fun and exciting world of microcontroller units (MCUs), plus the Beginner's All-purpose Symbolic Instruction Code (BASIC) programming language, and you! Most of us use a personal computer (PC) or some other iDevice (smartphone, tablet, digicam, et al), and they have an internal integrated circuit (IC) called a microprocessor unit (MPU) which is part the central processing unit (CPU) or the electronic "brains" of the device. An MPU acts as an interface with humans, and, on the other hand, an MCU is designed to act as an interface with other devices. MCUs are most often used to create artificial intelligence in machines giving them "self-awareness" and the ability to collect data at a distance via external sensors. This is called *telemetry* (Greek: remote or distant measuring), and the collection and dissemination of telemetry is what microcontrollers do best.

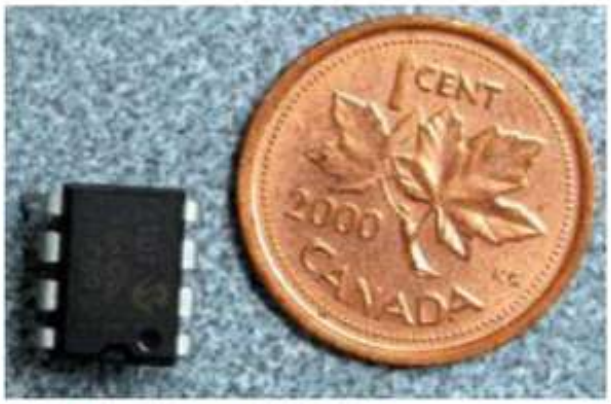
The MPU and MCU were created in 1971 (the Intel 4004 and Texas Instruments TMS 1000), and they control almost everything electronics or electrics we use today: computers, electric toothbrushes, microwaves, toasters, smartphones, digital cameras, remote controls, robots, washing machines, televisions, fridges, stoves, radios, dryers, alarm clocks, environmental and security control systems, sensors, medical devices, cars, planes, trains, satellites, GPS guidance systems, automatic doors, escalators, elevators, radios, etc., etc.

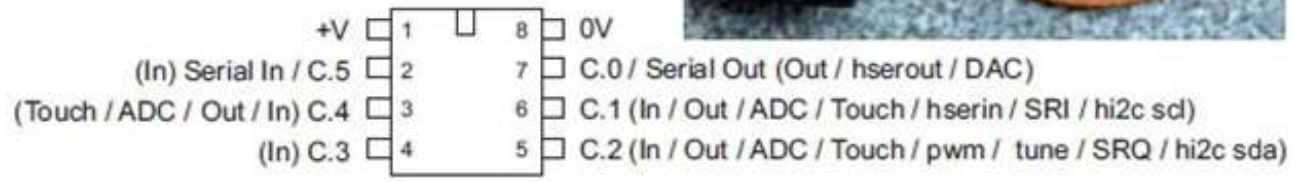
The line is often blurred between the two, and there is an overlap with many devices having both MPUs and MCUs, but there is one very important hardware difference between them: inside the MPU, program code and data both share the same memory space (called von Neumann architecture), while the MCU program code and data each have separate memories (called Harvard architecture and variations). This gives the MCU a tremendous advantage allowing it to work cheaper, faster, "harder", at a much slower clock speed, reducing power and cooling requirements. For example, some MPUs run so "hot" they can boil water, and they require heat sinks and cooling fans to vent heat which is wasted energy!

AN MCU BY ANY OTHER NAME

Popular MCUs used by hobbyists are: the Microchip PIC; Revolution Education's PICAXE (a PIC variant, Figure 1); the Parallax BASIC Stamp; the Atmel AVR, and the Arduino (an AVR variant), and we even have hybrid PIC/AVR designs! The majority of first-time MCU users should probably start with either the PICAXE or BASIC Stamp (my opinion) because they both use the BASIC language for programming, and have excellent educational kits designed for the classroom. *Note: Contrary to popular belief, the terms "PIC" and "AVR" mean nothing and are not acronyms for anything; they are just product trademarks. However, "PICAXE" may be a sly and subtle reference to the versatile pickaxe tool used by miners and others, but Rev. Ed. is mum on this. Regardless of what any manufacturer "christen" their product(s), they are all generically called "MCUs".*

FIGURE 1
8-pin layout of the PICAXE 08M2 MCU (quad-core parallel processor!) with size comparison to a penny.





The Revolution Education (Rev. Ed.) PICAXE is an inexpensive, general purpose, easy to use MCU platform for newbies, with a series of increasingly powerful chips starting with the 8-pin 08M2 (part of the "educational" line) and ending with the 40-pin 40X2 (part of the "advanced" line). You can download the free programming software (various systems supported) along with additional tools, examples, help files all from the Rev. Ed. website, plus they support a PICAXE user group and have a dedicated Amateur Radio forum on the website.

THE SCHOOLS EXPERIMENTERS KIT

Of all the Rev. Ed. products, the one I can highly recommend is the "Schools Experimenter Kit" or just "Schools Kit" (product no. AXE092, Figure 2 and Figure 3, next page); it uses a PICAXE 08M2 mounted in a small, self-contained experimental platform. This amazing little chip has built-in quad-core parallel processing ability (you can run 4 separate programs at the same time!), 2096 byes of [program] code memory, 256 bytes of data memory, 6 pins dedicated for input or output (digital or analog), along with a maximum clock speed of 32 MHz (default 4 MHz). Most MPU based iDevices require a lot of processing "horsepower", battery power, and human control "at the wheel" because people want to watch videos, play games, burn discs, text, or download and store gigabytes of "stuff", otherwise you just have a very expensive paperweight that can't really do much without you, but MCUs, like the PICAXE, are designed to do their jobs as succinctly and efficiently as possible without needing us at all!

You may also want to get a copy of David Lincoln's, *Programming and Customizing the PICAXE Microcontroller* because there's an excellent chapter on the Schools Kit along with various projects. Another excellent, but more technical book, is Ron Hackett's *PICAXE Microcontroller Projects for the Evil Genius.* Ron also has a web store supporting the various book projects.

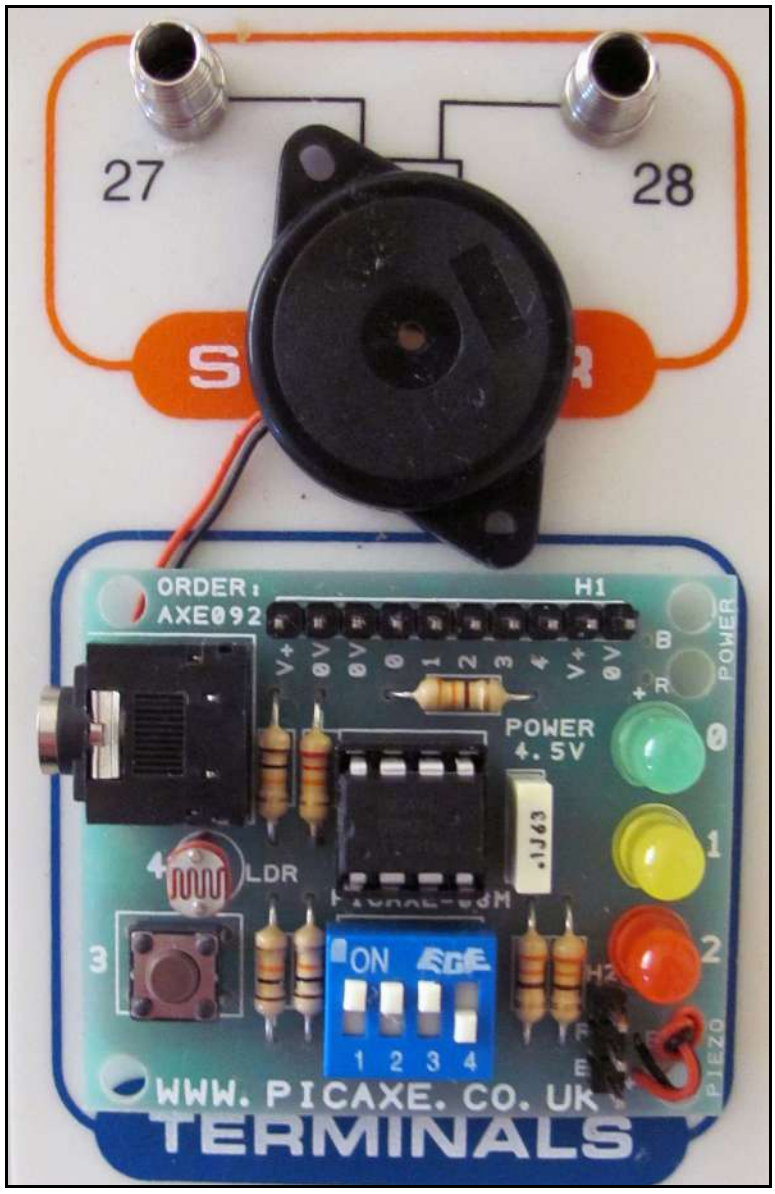


FIGURE 2 (RIGHT)

Rev. Ed. Schools Experimenter Kit mounted on my old 300-in-1 electronics kit. You can create programs to control 3 light emitting diodes (LEDs), a push-button switch, piezo speaker, and light dependent resistor (LDR). The programming (and also serial data output) is via a stereo phone jack. The kit comes with everything except for the headers (you can use male or female) and the programming cable. *Note: I reversed the positions of the green and red LEDs because it made more sense to wire it to the piezo speaker and create "red alert" alarms.*

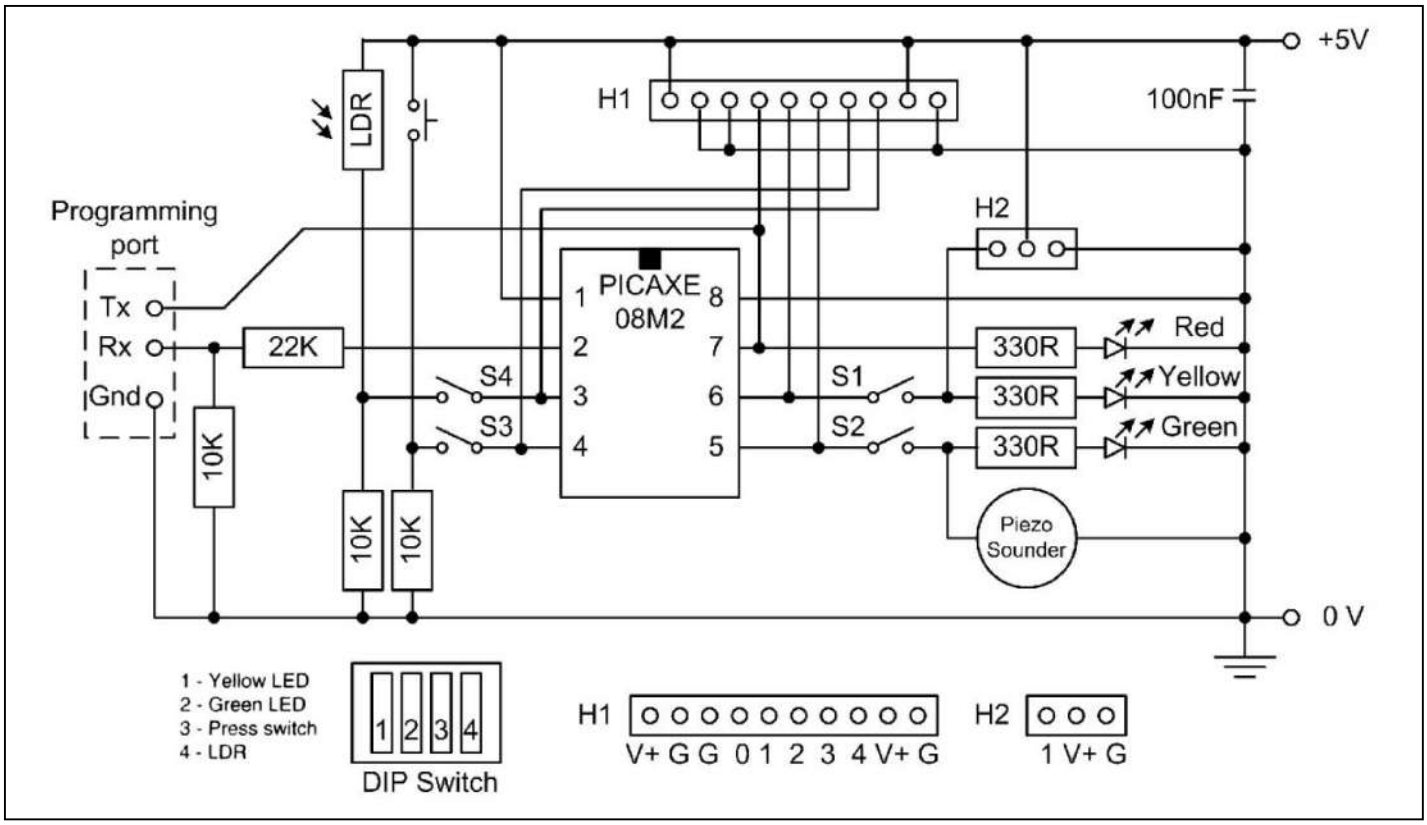


FIGURE 3: Schools Kit Schematic

YOUR FIRST PICAXE PROGRAM!

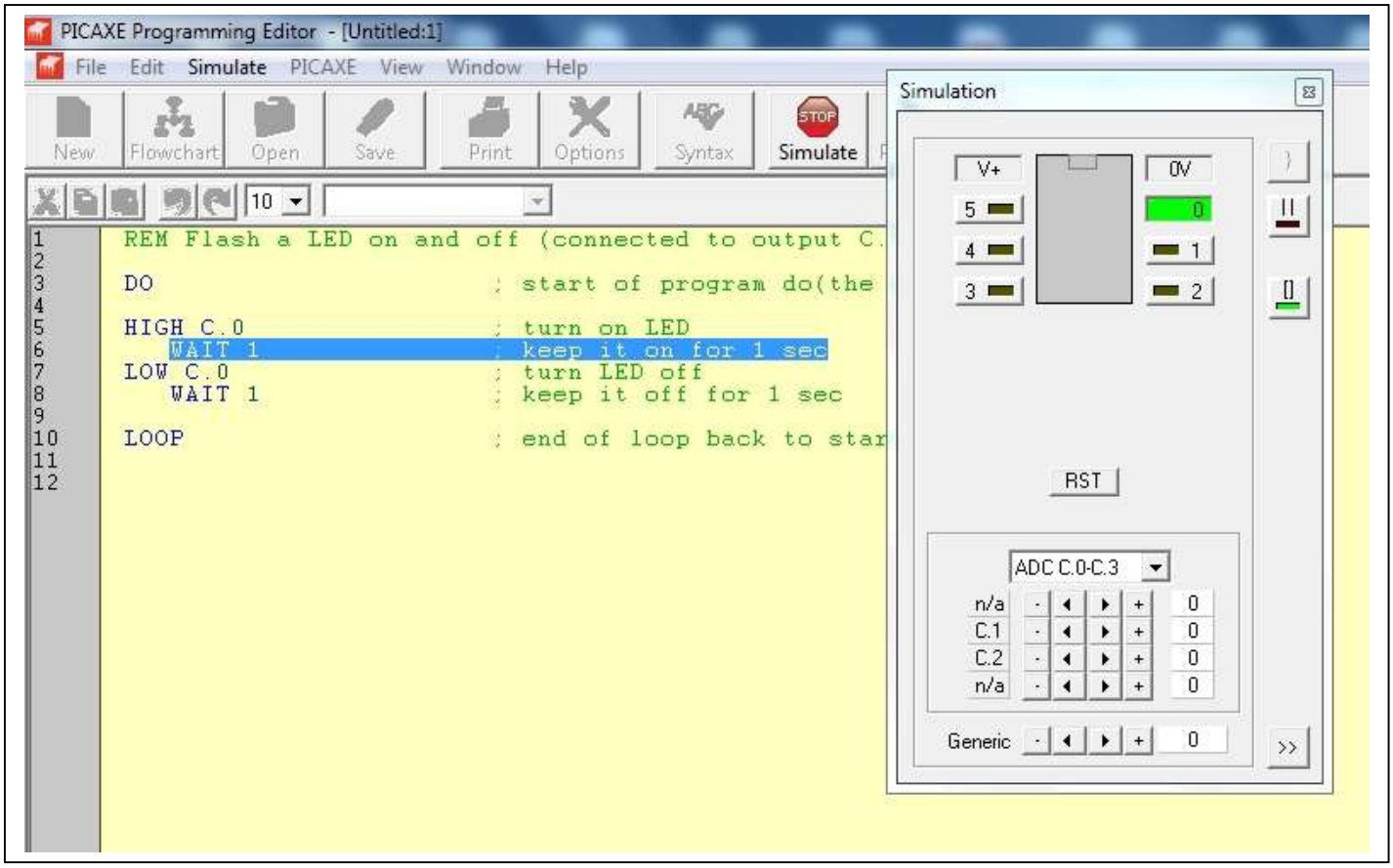
By tradition, the first computer program you write is called "Hello World!" In this case, we'll blink one of the Schools Kit's LEDs. You'll need to build or buy proper programming cable (USB or serial), a Schools Kit, and download and install the free PICAXE Programming Editor from the Rev. Ed. website. When power is applied to the PICAXE the program will run automatically, and the red LED will blink (unless you switched the green/red LEDs as I did!).

```
REM Flash a LED on and off (connected to programming "port" C.0, physical pin 7)

DO                  ; start of program do (the following code…)

  HIGH C.0          ; turn LED on (logical high or 5V DC)
    WAIT 1          ; keep it on for 1 sec
  LOW C.0           ; turn LED off (logical low or 0V DC)
    WAIT 1          ; keep it off for 1 sec

LOOP                ; end of loop …go back to start …loop until power removed
```

The BASIC program is called the “source” code and it’s automatically converted into binary code (0’s and 1’s) to become the “executable” code which is then loaded or “burned” into the MCU and executed by the MCU. Notice the source code is very English-like with user indentation (recommended to offset different “sentences” and “paragraphs”) and comments to make it easier to read the “story”. This is done because others may want to made changes or additions, and comments help them “get inside your head”, this is especially essential for large and complex programs with hundreds or thousands of lines of source code!

A really neat feature of the Programming Editor lets you execute the program using the virtual circuit simulator (Figure 4) to see if what you want to happen will happen—without building the actual circuit!

FIGURE 4 Programming Editor Running Software Simulation



MY FINAL

In the next column, I’ll continue on with MCUs and introduce the Parallax BASIC Stamp. The company has designed an excellent electronics and robotics training package (the Boe Bot) which is quite popular in the U.S. school system. For now, we won’t cover the Atmel or Arduino platforms since they use different (and more difficult to learn, for some) programming languages.—73

REFERENCES AND RESOURCES

Revolution Education http://www.picaxe.com

Schools Experimenter Kit http://www.picaxe.com/docs/axe092.pdf

PICAXE Microcontroller Projects for the Evil Genius
http://jrhackett.net

Microcontrollers and Amateur Radio
http://www.qcwa70.org/presentations1.htm

Programming and Customizing the PICAXE Microcontroller

SOME CANADIAN MCU & ELECTRONIC SUPPLIERS

Cana Kits http://www.canakit.com
Robot Shop http://www.robotshop.com/ca
SOLARBOTICS http://www.solarbotics.com

ADDITIONAL ARTICLES

Microcontroller Projects for the Radio Amateur *TCA, May-June 2011, (p. 20-23).*

Introducing Microcontrollers to Your Amateur Radio Club *TCA, Jan-Feb 2010, (p. 43-47).*

Canadian Clubs Dig PICAXE *QST, Apr 2011, (p. 61-63).*

Infrared Radio Control for your HF Transceiver *QST, Nov 2010, (p. 36-39).*