Bury the hatchet, unbury the axe

Getting started with PICAXE

By **Wouter Spruit** (NL)

Microcontrollers can be used to add processing capabilities to any project, and off-the shelf microcontrollers are often used to fulfill a specific purpose. Programming a microcontroller with your own firmware gives you complete freedom to implement smart input and output for your projects. The PICAXE system gives any hobbyist the chance to program a microcontroller and interface it to peripherals with little hassle and at low cost.

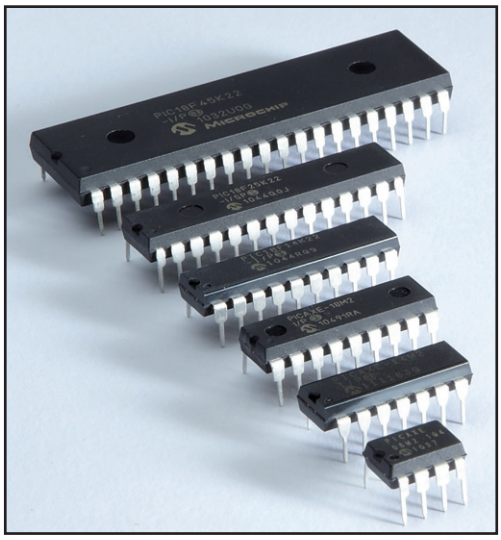

Figure 1.
PICAXE chip sizes.

What? Why? How?

The PICAXE products are a range of Microchip PIC microcontrollers programmed with a special firmware by Revolution Education [1]. The philosophy behind the PICAXE system is that the best way to learn how to program and implement microcontrollers in a project is to use a cheap system that doesn't require a lot of experience but still offers complex interfacing capabilities that make the microcontrollers suitable for more advanced projects. The PICAXE line offers a range of microcontrollers of different shapes and sizes, from simple microcontrollers with only 8 pins and limited capabilities, to the more advanced 40-pin 40X2 model (**Figure 1**). Since the introduction of the PICAXE system over 15 years ago, more recent parts have replaced the PIC microcontrollers that form the foundation of the PICAXE system, offering not only more memory and computational power, but also parallel execution of tasks in many cases [2].

The Revolution Eduction webstore [3] not only offers the PICAXE microcontrollers, but also a range of accessories: from components, breadboards, cams and gears to cables, I2C peripherals, LCD screens and starter packs. The website aims to offer everything necessary to get started programming the PICAXE right away. The online store even offers kits to build toys ~~like~~ such as robots based on PICAXE microcontrollers. All peripherals for sale come with extensive documentation on how to interface them with a (PICAXE) microcontroller. So even if you're not using a PICAXE microcontroller in your project, checking out this store for components might be worthwhile.

The secret that defines PICAXE is the special firmware. A PICAXE chip is a Microchip PIC pre-programmed with the PICAXE firmware, including a BASIC interpreter and a number of pre-loaded functions. The PICAXE code allows a user to upload a PICAXE BASIC program at any time, via the serial connection (as long as it's not in use by the current programming). The new code starts executing right away. Because the PICAXE programs are interpreted BASIC code, the number of instruc-

tions carried out per second are relatively limited, rendering bit-banged communications to interfaced devices infeasible. However, the pre-programmed functions implement some common bit-banged input/output functionality, for instance enabling the PICAXE chip to communicate with I2C peripherals or sending and interpreting IR signals. The advantages of using interpreted BASIC code for writing PICAXE programs is the way the programs are uploaded to the device. The interface required to program regular Microchip PICs is complex and hard to build and use, in comparison to the PICAXE programming interface. PICAXE projects commonly include a jack connected to the chip's serial interface pins, which allows for quick and easy reprogramming of the chip in the field. Just plug in the power and connect the serial jack to a PC and you're ready to upload a program to your PICAXE chip.

Though the PICAXE BASIC programs can be written and uploaded in a very short time, complex mathematical operations requiring thousands of clock cycles per calculation are best not implemented using the real-time interpreted BASIC code. The code would run too slowly to be of any use in real-time sensitive projects.

Alternatives to PICAXE

Regular Microchip PICs require a PIC programmer to upload new program code to the device. Though projects such as the "el cheapo" [4] can be used to decrease the price of obtaining a PIC programmer, the cost and complexity of starting a project to simply blink an LED on a fresh PIC can be fairly intimidating to those just starting to get interested in programming their own microcontroller. The same goes for the Atmel AVR microcontrollers. Though the Arduino project [5], consisting of a Atmel AVR microcontroller on an interfacing board together with an IDE and documentation, also offers an easy way to get started with microcontrollers, it's more expensive and difficult to implement in actual projects in comparison to PICAXE.

Support and documentation

Because of the educational nature of the PICAXE products, it's no surprise extensive documentation is available. The PICAXE manual is included with the programming editor and AXEpad, in addition to being available on the PICAXE website [6]. The manual is split in three parts, called "getting started", "BASIC commands" and "microcontroller interfacing circuits." The first part contains information on the technical capabilities and pinouts of various PICAXE products, instructions on powering and connecting to the device, as well as a number of tutorials to get started with the system. The second part of the manual covers descriptions for all BASIC commands, as well as providing information on the command availability on the various PICAXE chips and programming examples. The third part of the manual is a collection of interfacing examples. The manual not only covers interfacing basic components such as a switch, LED or motor via Darlington pair, but it also provides examples for interfacing the PICAXE chip with commercially available components and chips, including instructions for adding an LCD screen via I2C, and it even covers connecting the PICAXE to a PC via a serial connection. The best part about the examples is that they don't apply exclusively to interfacing PICAXE with components, but the general principle applies to interfacing peripherals with any non-PICAXE chip (still, it might be wise to consider the chip's operating voltage and maximum current it could sink and source).

Though it was originally designed for educational purposes, PICAXE is also especially suitable for use in many hobbyist projects. PICAXE users are encouraged to upload their projects to the PICAXE website. The gallery of user projects has become a great showcase for PICAXE capabilities. The forums available on the website are a good place to get answers to your questions regarding the PICAXE system or interfacing peripherals for your electronics projects.

Supporting software

The PICAXE system allows the user to upload BASIC programs from a PC to a PICAXE chip via a serial connection. PICAXE comes with a feature-rich development environment for Windows called the programming editor, allowing the user to quickly connect to, program and debug PICAXE chips. A cross-platform alternative to the programming editor is available under the name AXEpad. AXEpad provides the most important features of the

Figure 2.
USB-to-jack cable.

programming editor, including: editing and uploading code to the PICAXE chip, a terminal window, and even several code generation wizards. In comparison to the programming editor, AXEpad does lack some features, for instance the on-screen tool to test and step through a simulation of a PICAXE program.

To use the editor, connect a PICAXE chip using either the serial-to-jack or the USB-to-jack cables (see **Figure 2**) and specify the serial port the programming editor (or AXEpad) should use to connect to the chip. To tell the editor what kind of chip it is programming, the PICAXE type is selected manually. The program also offers a function to look up the firmware version for a connected PICAXE chip. Once the chip is connected, it is programmed by uploading a written/loaded a BASIC program from the editor directly to the PICAXE chip. The new program starts running on the PICAXE chip right away.

Blinking an LED

This simple example will show how to program a PICAXE chip to blink an LED. A PICAXE 08M is used in the example circuits, but the general principle applies to every PICAXE chip. Use the pinout diagrams [7] corresponding to your PICAXE chip to select an output to connect an LED to. In the case of the PICAXE 08M, pin 3 (C.4) was selected from the pinout of the PICAXE 08M2 (as depicted in **Figure 3**). Though Revolution Education offers starter packs that already include all necessary interfacing electronics, this example will show how to connect a PICAXE chip ~~yourself,~~ on a breadboard. The pins relevant to our setup are pins +V (1), 0 V-(8), serial in (2), serial out (7) and the output pin that connects to the LED (3).

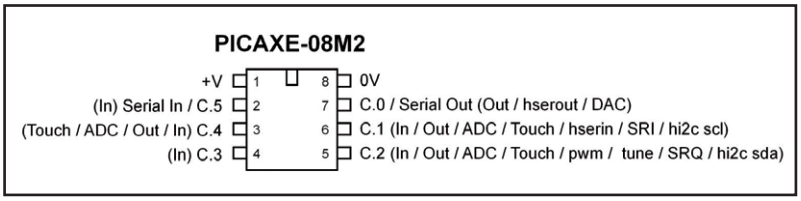

Figure 3.
Pinout of the PICAXE 08M2.

The blinking LED circuit is shown in **Figure 4**. The power pins +V and 0V are connected to a DC power supply from 3.0 V to 5.0 V. To use one of the battery packs (for instance, from

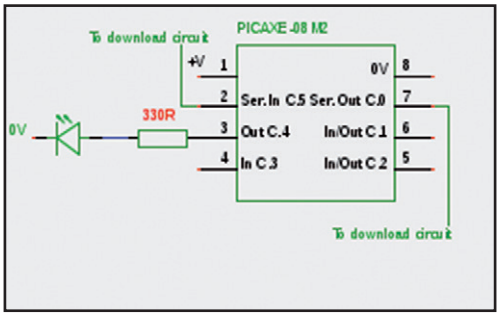

Figure 4.
Blinking LED circuit.

the Revolution Education store), take care to use no more than three 1.5 V AA batteries or four 1.2 V AA rechargeable batteries. The download circuit according to **Figure 5** is connected to the serial in, serial out and 0 V pins on the PICAXE. The LED is connected in series with a 330 Ω resistor from pin 3 to ground. When the download cable is not connected to a homemade circuit, the serial in pin (2) must be pulled low by tying it to 0V with a 33k resistor.

Programming the PICAXE

Install either AXEpad for GNU/Linux and Mac, or the programming editor for Windows [8]. If you're using the USB to serial jack cable, some operating systems require USB to serial drivers before the cable can be used [9]. For this example, the author is using AXEpad on GNU/Linux, with an RS-232 serial cable connected via 3.5-mm jack to a PICAXE 08M on a solderless breadboard. Start the software a few seconds after the serial cable/USB cable has been connected to a powered PICAXE chip. The correct PICAXE chip version and COM port have to be specified before a program can be uploaded. For AXEpad, select the correct serial port in View → Options →Tab:Port (if you're using the USB-to-serial cable, it will be listed too). Test the connection by clicking the Firmware button in the mode tab of the View → Options dialog.

After everything is set up correctly, the firmware button on the mode tab returns the type of chip that is currently connected. If this type is not selected on the same tab, do so now. Now we are ready to write our first PICAXE program. Enter the code in **Listing 1** in the editor. A note on the pin numbering: the internal names for the pins can be confusing. Some chips can have multiple internal IO blocks, requiring a prefix character to show which block to select. In this example, the author used the older 08M chip, and in this version, no prefix character was needed (for output 4 on pin 3: "high 4") . If the code from Listing 1 is giving you problems with your version of the PICAXE chip, try adding the prefix character according to your ~~chips~~ chip's pinout (for instance for pin 3, corresponding to output 4 in a C block, "high C.4"). Now upload the program to the PICAXE by clicking the upload button, selecting the PICAXE->program option or by pressing F5. The PC interface shows the uploading progress and the memory available on the PICAXE chip. Once it's done uploading, the "programming successful" message shows that the uploading process went OK. The new program starts running on the PICAXE instantly, and the LED should start flashing.

Listing 1: our very first PICAXE program

```
do       ;repeat forever
  high 4     ;set output 4 high
  pause 1000   ;wait 1 second
  low 4    ;set output 4 low
  pause 1000     ;wait 1 second
loop
```

Listing 2: LED button switch

```
do       ;repeat forever
  if pin3=0 then  ;if the button is pressed
    high 4   ;light LED
  else     ;if the button is NOT pressed
    low 4  ;turn off LED
  endif    ;closes if statement
loop
```

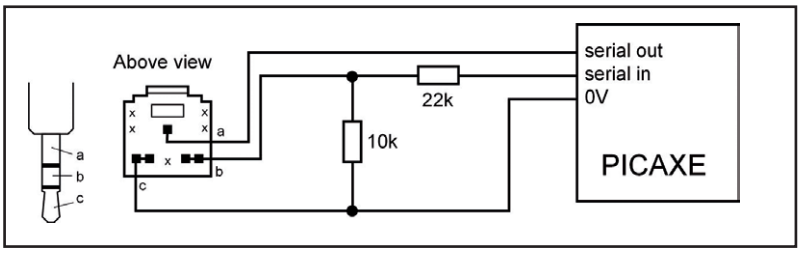

Figure 5.
Download circuit.

Adding a push button

The next example shows how to make the

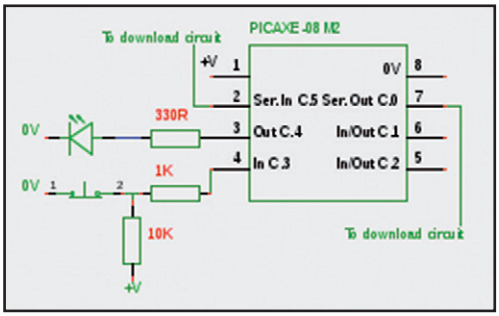

Figure 6.
Push button circuit.

PICAXE light an LED when a button is pushed. Connect a push button to pin 4 of the PICAXE chip (it's called input 3 in the software), as depicted in **Figure 6**. Use the code from **Listing 2** to program the PICAXE chip. After uploading the code, the LED only lights when the button is pushed down. The "if" statement in the code checks if output 3 (on pin 4) is pulled low by a closed switch. If so, it lights the LED; if not, it turns the LED off. **Figure 7** shows a breadboard version of a PICAXE 08M programmed with the code from Listing 2, disconnected from the programmer circuit.

The examples have shown just how easy getting started with PICAXE can be. It's not hard to see how the range of components available in the tech supplies web shop, together with the extensive documentation provided by Revolution Education, inspired many hobbyists to start programming PICAXE microcontrollers for their electronics projects.

(130137)

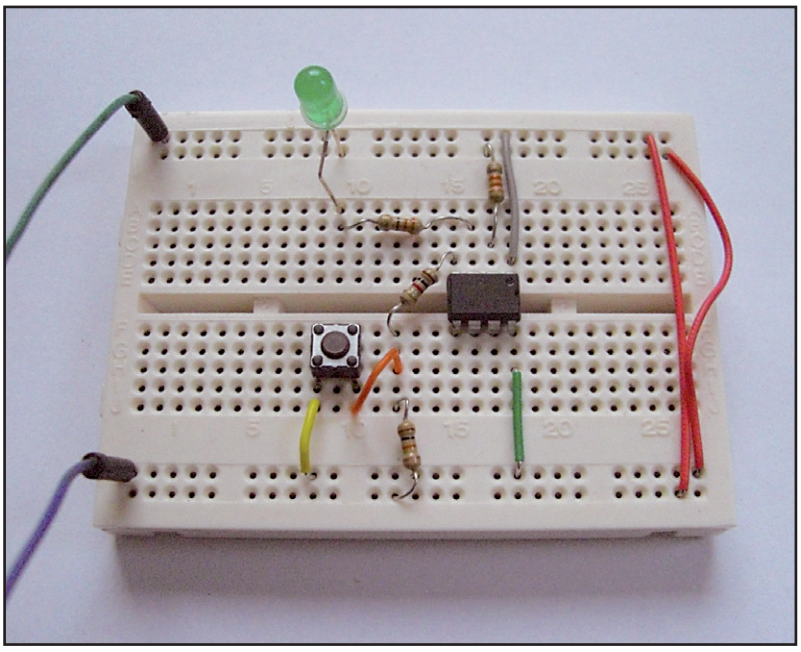

Figure 7. Push button setup ready with a PICAXE 08M in a breadboard.

Internet Links

[1] www.picaxe.com

[2] www.picaxe.com/What-is-PICAXE/PICAXE-Chip-Sizes

[3] www.techsupplies.co.uk/PICAXE

[4] www.rentron.com/myke4.htm

[5] www.arduino.cc

[6] www.picaxe.com/Getting-Started/PICAXE-Manuals

[7] www.picaxe.com/What-is-PICAXE/PICAXE-Pinouts

[8] www.picaxe.com/Software

[9] www.picaxe.com/Getting-Started/Driver-Installation