24.13 The ID-O-Matic Station Identification Timer

Would you rather not watch the clock when engaged in a QSO? Repeaters usually have an automatic 10-minute ID to remind you when it's time to identify your station, but on HF and simplex it's more of a problem. Dale Botkin, NØXAS, set out to design a simple 10-minute ID timer with a reminder beep and some sort of visual indicator, something along the lines of the ID timer portion of the old Heathkit SB-630. Along the way he added a few features to make the timer more useful, and as sometimes happens, things just kind of snowballed from there. The ID-O-Matic described here and shown in **Fig 24.54** is an automatic ID timer/annunciator with a rich set of features that make it useful for many applications.

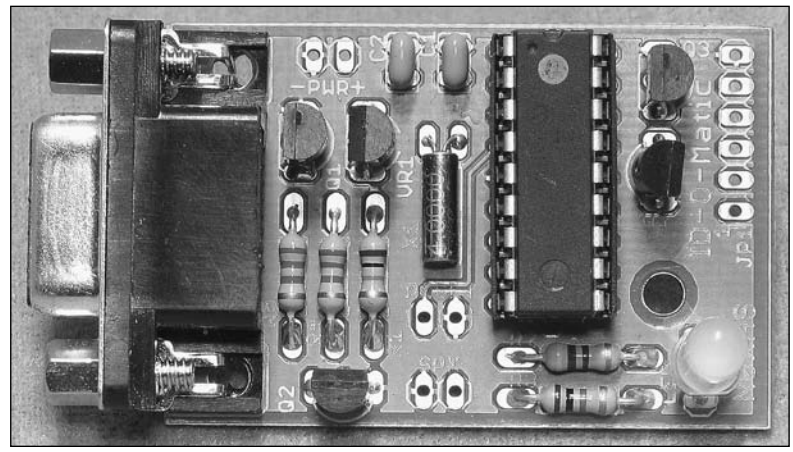

Fig 24.54 — This version of the ID-O-Matic is built from a kit offered by www.hamgadgets.com. The parts count is low, and it can easily be built on a prototype board.

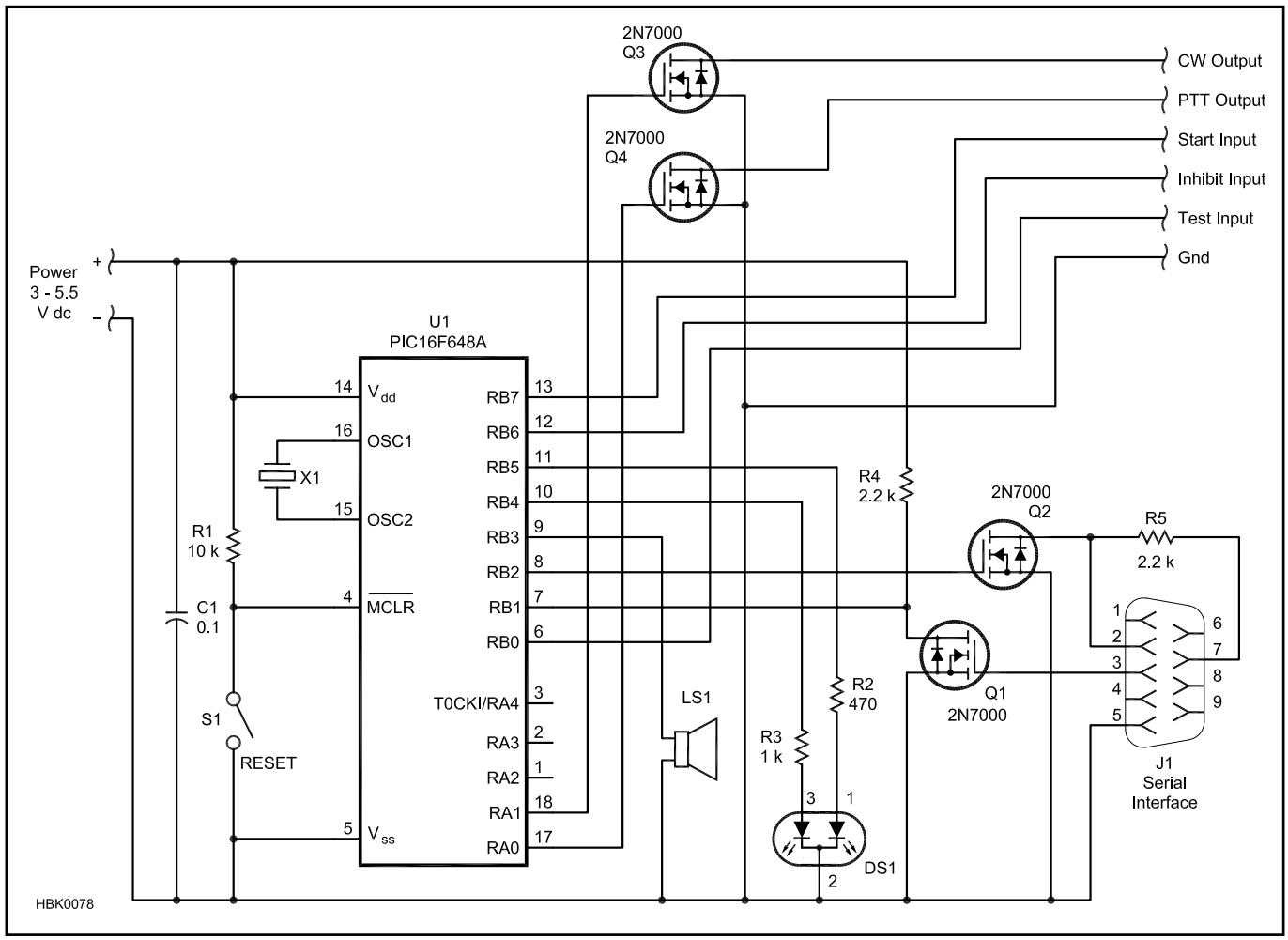

Fig 24.55 — Schematic of the ID-O-Matic. Resistors are ¼ W.

DS1 — Dual-color LED, MV5437.
J1 — DB9 female connector (Mouser 152-3409).
LS1 — Soberton GT-111P speaker or equiv (Mouser 665-AT-10Z or Digi-Key 433-1020-ND).
Q1-Q4 — 2N7000 MOSFET.
U1 — Microchip PIC16F648A. Must be programmed before use (see text). Programmed chips and parts kits are available from www.hamgadgets.com. (ARRL now publishes a book about PIC Programming for beginners.)
X1 — 4 MHz cylindrical crystal (Mouser 520-ECS-400-18-10).
Enclosure: Pac-Tec HMW-ET (Mouser 76688-510).

FEATURE HIGHLIGHTS

There's an old saying that goes, "When all you have is a hammer, everything looks like a nail." While the author has a pretty well stocked junk box, he never liked building 555 timers and the like. He tends to use whatever is handy and can cobble together with an absolute minimum of parts, and likes to be able to make changes or add features later on. That often means writing firmware for a PIC microcontroller, and this time was no exception.

The basic device uses a single PIC 16F648A processor selected for its memory capacity, internal oscillator, hardware USART and other features. All features are implemented in program code, with just enough external hardware to provide a useful interface to the user. Audio and visual output is used to indicate the end of the time period, and one user control is all that is required other than a power switch.

Let's take a quick look first at the basic features. In its most basic timer mode, a single pushbutton is used to start the timing cycle by resetting the PIC. This starts a 10-minute timer running and lights the green section of a red-green dual color LED. The LED remains green until 60 seconds before the set period expires. With less than 60 seconds remaining, the LED turns yellow; this is accomplished by illuminating both the green and red halves. At 30 seconds the LED begins to blink, alternating between yellow and red. When the time period expires, the LED turns red and the timer starts beeping until you reset it with a pushbutton switch. Now the cycle starts again, ready to remind you in another 10 minutes.

So far so good, but using a PIC like this is a little bit of overkill. Of course no good project is complete until "feature creep" sets in! With the timing and audio portions of the program done, some other features were pretty simple to add. What if you want to ID at some other interval instead of 10 minutes, or have the device automatically ID in Morse code? What if you want to use it for a beacon or a repeater ID device? How about CW and PTT keying outputs for a fox hunt beacon?

Eventually the firmware evolved to have a fairly robust set of features. There is a serial interface for setup using a computer or terminal. You can set the timeout period anywhere from 1 to 32,767 seconds. You can set select a simple beeping alert, or set a Morse ID message with up to 60 characters. For repeater and fox hunt use, there are outputs for PTT and CW keying just in case they might be needed, as well as a couple of inputs intended for COR or squelch inputs if the chip is used in repeater mode. We'll cover all of that in a bit; first, let's look at the basic functions and the hardware design.

CIRCUIT DESIGN

As shown in **Fig 24.55**, the hardware is quite simple. In its basic form all you need is the PIC, a red and green dual-color LED, and a few other parts as seen in the schematic. Supply voltage can be 3 to 5.5 V, so you can use three AA alkaline cells. You can also add a 5 V regulator, allowing the use of a 12 V power supply. Despite their size, 9 V batteries have surprisingly low capacity and will only last a few days of continuous use in a project like this.

R1 is a pull-up resistor whose value is not critical; 10 kΩ is a good value. Its function is to hold the !RESET line high until S1 is pressed. R2 and R3 set the current for the red and green sections of the dual LED. They are different values because the red section of the LED is usually substantially brighter than the green. R4 pulls the serial RxD line high when no serial connection is present. For use in the shack to remind you to ID every 10 minutes, that's all you need.

The original prototype fits in a Pac-Tec enclosure model HMW-ET. For audio output, try a tiny Soberton GT-111P speaker. About the size of a small piezo element at 12 mm diameter and about 8 mm tall, this is actually a low current magnetic speaker that can be driven directly from the PIC's output pinch Switches can be whatever style you like. Many of the parts can be obtained at a local RadioShack, where you may also find a suitable enclosure and battery holder.

A level converter circuit allows you to connect a PC or terminal to set up some of the more advanced features. A single chip solution such as a MAX232 or similar chip could be used, but the circuit shown is effective, very low cost and easy to build. Most modern serial ports do not require full EIA-232 compliance, but will work fine without a negative voltage for the interface lines. Accordingly, the interface presented in the schematic simply inverts the polarity of the signals and also converts the ±12 V logic levels that may be present on the PC interface to levels appropriate for the PIC I/O pins. By using MOSFETs instead of the more traditional NPN and PNP transistors we can eliminate some of the parts usually seen in level shifting circuits like this. The only additional components are Q1, Q2 and R4, a current limiting resistor for the TxD line.

With a terminal or terminal emulator program you can set your own delay time, CW speed, ID message and select a simple beep or a CW ID. Inputs used for repeater operation (!INHIBIT and !START) are an exercise left to the builder. The thing to remember is that these inputs are active low, and the input voltage must be limited to no more than the PIC's V_{dd} supply voltage.

PUTTING THE ID-O-MATIC TO WORK

When power is applied, the PIC is set to beep at the end of a 10-minute time-out period. After a series of reminder beeps, it will automatically reset and begin a new timing cycle. To change any of the settings, simply connect a serial cable to your PC or a dumb terminal and the ID-O-Matic. Any serial communication program such as *PuTTY*, *HyperTerminal* or *Minicom* can be used. The communication settings are 9600 baud, 8 bits, no parity and no handshaking. Hit the ENTER key twice to view and edit the configuration. The program will step through a series of prompts as shown in **Fig 24.56**. At each prompt you may either enter a new setting or simply hit ENTER to keep the existing setting, shown in parenthesis.

There are a few optional inputs and outputs that could be used to make the timer suitable for use in a "fox" transmitter or repeater.

- Pin 6 (RB0) is the !TEST input. You can momentarily ground this input to hear your ID and/or beacon messages. If no messages have been stored, the ID-O-Matic will announce its firmware version. During normal operation, this pin will select the alternate ID message. This may be useful to indicate, for example, if a repeater site has switched to battery power, or if a "fox" transmitter has been located.
- Pin 12 (RB6) is the !INHIBIT input. This pin is normally held high by the PIC's internal weak pull-up resistors, and can be driven low by some external signal (squelch, for example) to delay the CW ID until the input goes high.
- Pin 13 (RB7) is the !START input, used only in repeater mode. Also pulled high internally, a LOW logic signal on this pin will start the timer. This can be handy to have your rig or repeater ID 10 minutes after a transmission, but not every 10 minutes. In repeater mode, the chip will ID a few seconds after the first

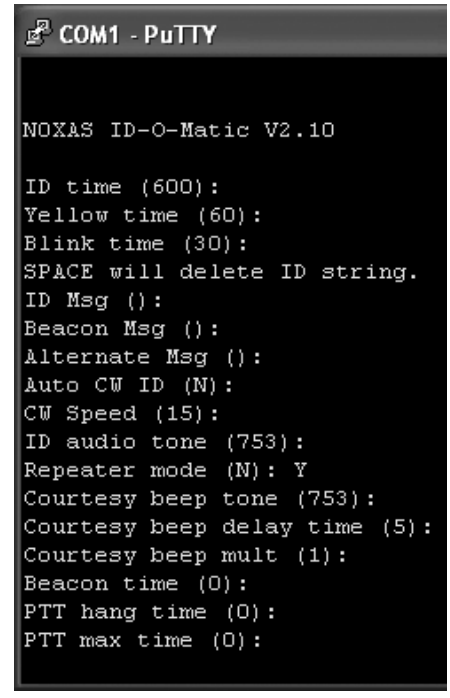

Fig 24.56 — Using *HyperTerminal* to set up the ID-O-Matic

low signal on the !START input. If repeated !START inputs are seen, subsequent IDs will occur at the programmed intervals.

• Pin 17 (RA0) is a PTT output. PTT goes high about 100 milliseconds before the audio starts and stays high until 100 ms after the end of the message. With the 2N7000, the PTT and CW outputs can handle up to 60 V at 200 mA.

• Pin 18 (RA1) is a CW output pin; it will output the same message as the audio output.

The original program code took several days to write and tweak, and has undergone numerous revisions since then. As with any project like this, the testing was the time consuming part. The author used the PIC C compiler from CCS, Inc. to write the program, but the logic is simple enough that porting to assembly, BASIC or a different C compiler should be pretty straightforward.

The code is relatively easy to adapt to your own equipment or requirements. For example, you may wish to move certain functions to different pins, change the polarity of various signals or change some of the timing defaults. The .LST file contains the C source with the ASM equivalents, and is a good starting point for someone wishing to understand the program logic in assembly rather than C. See the *ARRL Handbook CD* at the back of this book for these files.

The really interesting thing about this project is its versatility. Think of the ID-O-Matic as a general purpose PIC based project platform. With a serial interface, a few available inputs and outputs and a dual-color LED, the hardware can be adapted to many different uses simply by changing the PIC's program code. For example, the author built a 40 yard dash timer for his son's high school football team by substituting an infrared LED for the speaker, connecting an IR detector to one of the inputs and using another for a pushbutton switch. The serial interface is connected to a serial LCD display module. Within a couple of days he had a working device, customized with his son's team name, that can be used to time various events.

It's a good little project for beginning PIC users, and a useful station accessory that can be built in an evening and carried around in a shirt pocket.