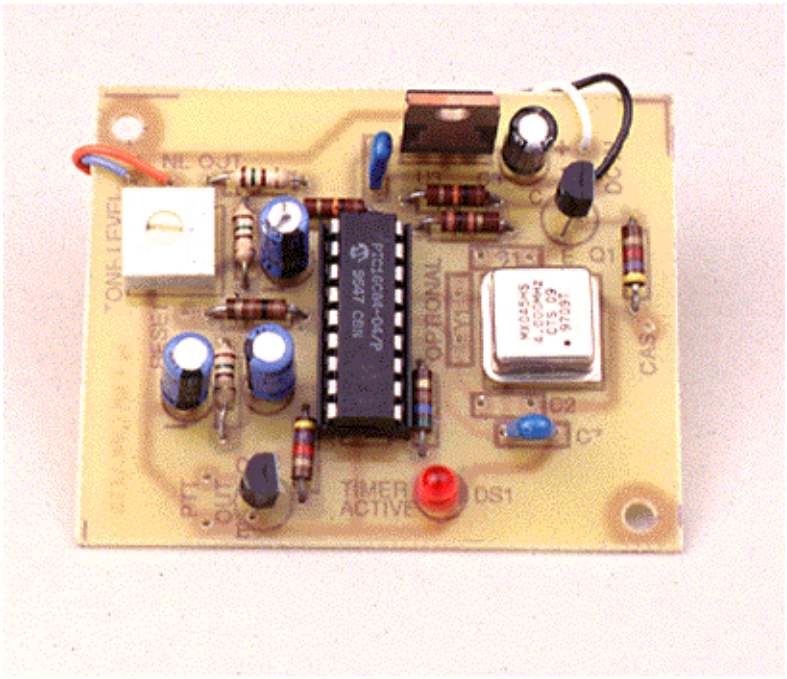
A PIC of an IDer

Here’s a useful project—and some tips on how you can get started programming your own PICs!

By Bob Anding, AA5OY



Over the past few years, I’ve designed and built several IDer circuits. The first was a diode array that I installed in my repeaters and used for years—it’s now in my junk box. Next came some three and four-chip EPROM-based IDers. The four-chip version is currently running in several repeater systems around the New Orleans area.

Not long ago, while I was talking with several other repeater owners, we kicked around the idea of building a small, low-power IDer using a minimum number of chips. We wanted to use the IDer for emergency repeaters and "fox-hunting" service (battery powered). One of Microchip’s PIC ICs seemed ideal for this application. [1]

One IC Does It All

The PIC16C84 is a relatively recent addition to Microchip Technology Inc’s range of microcontrollers. [2] It’s easy to work with because its program memory uses EEPROM technology. During a project’s prototyping stages, this gives the 16C84 an advantage over standard EPROM microcontrollers (including other PICs) because it can be reprogrammed in a few seconds. (Microchip’s 16F84 uses flash memory technology and is directly interchangeable with the 16C84.)

This single-PIC16C84 IDer is about as small as it gets. The circuit is simple because the software does most of the work! At 5 V, the IDer draws only 6.5 mA in the wait state, and the program will remain in the IC’s memory for years ("> 40 years" according to Microchip.—*Ed.*) Parts for this project are available from Digi-Key. *Programmed* ICs and PC boards are available to make things easy for you, too. [3] For more fun, adventure and learning, you can even program your own PIC—see the sidebars "**Programming Your Own PIC**" and "**Further Reading**." Source code for this project is available for downloading, too. [4] I have found programming PIC chips to be fun and challenging! The Morse code encoding scheme I used for this project is presented in the appropriately titled **sidebar**.

Circuit Description

Refer to **Figure 1**. U3 supplies a +5 V regulated source for the rest of the circuit. The output of clock module U2 feeds the clock input of U1 (pin 16). U1 pin 18 is held high by R9. When the receiver's CAS signal pulls pin 18 low, U1's timer starts its 10-minute countdown. When the countdown is complete, U1 pin 8 goes high and provides a dc level (+5 V, 20 mA) for the transmitter's PTT circuit. After a short programmed delay, U1 outputs Morse code tones at pin 7. The tone frequency is set at 446 Hz. An optional TIMER ACTIVE LED output is available at pin 13. When the timer is active, the LED attached to pin 13 blinks at a 2.3 sec rate. (There is an easy way to change the timer length—and the delay between the action at U1 pins 8 and 7—from within the software; see **Note 4**).

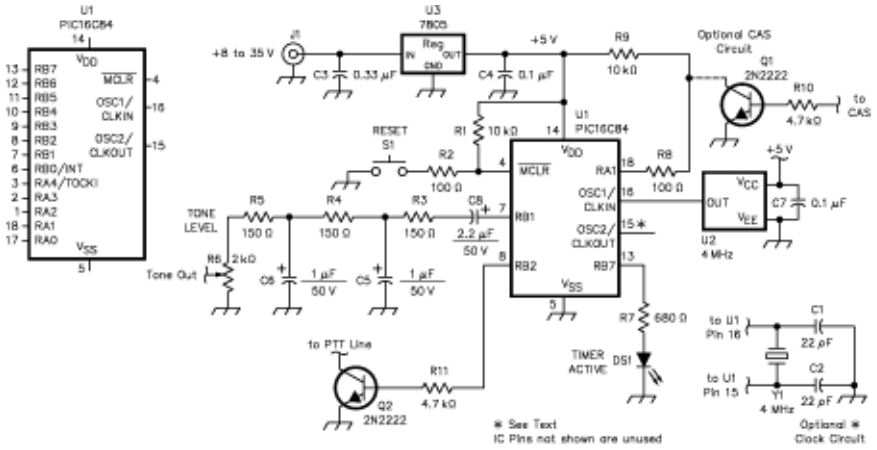


Figure 1—Schematic of the IDer. Q2 keys positive PTT lines. S1 is an optional RESET switch. Unless otherwise specified, resistors are 1/4 W, 5% tolerance carbon-composition or film units. Part numbers in parentheses are Digi-Key; equivalent parts can be substituted. (Digi-Key Corp, 701 Brooks Ave S, Thief River Falls, MN 56701-0677 tel 800-344-4539, 218-681-6674, fax 218-681-3380; http://www.digikey.com.)

C1, C2—22 pF disc ceramic (DK P4841); optional, see text.

C5, C6—1 µF, 50 V aluminum electrolytic (DK P5268)

C8—2.2 µF, 50 V aluminum electrolytic (DK P5269)

DS1—LED (DK LT1136)

J1—Coaxial power jack (builder's choice)

Q1, Q2—2N2222 NPN transistor (DK 2N2222)

R6—2 kΩ potentiometer (DK 3362P-202)

S1—Momentary push-button switch (DK CKN9015); optional

U1—*Programmed* PIC16C84-04/P or PIC16F84 -04/P; see text and Note 5. (DK PIC16C85-04/P or PIC16F84-04 /P; these are *unprogrammed* ICs.)

U2—4 MHz clock oscillator (DK CTX155)

U3—7805 or LM340T 5 V, 1 A positive voltage regulator (DK LM340T)

Y1—4 MHz crystal (DK CTX006); optional; see text.

Misc: Enclosure, 18-pin IC socket, wire, solder, hardware

Instead of the clock module, you can use a 4-MHz crystal as shown in the inset of **Figure 1**. (Someone once told me to stick with clock modules...they always start on time...) In any case, stay with the 4 MHz clock frequency: All of the program's timing cycles are based on using a 4 MHz clock.

S1 can optionally be used to reset the timer at any point in the countdown. R6 (**TONE LEVEL**) provides a means of adjusting the amplitude of the Morse tones.

Construction

There's not much to say here. Obviously, you can approach building this project in any of several ways, including perfboard and point-to-point connections, or a PC board (see **Note 3**). For my prototype, I used one of Radio Shack's circuit boards (276-148).

All computer chips generate some RFI. Following the usual construction precautions will prevent most, if not all, problems: A good metal box and some feedthrough capacitors should work fine.

Summary

Using the PIC chip has been helpful in simplifying my designs. I hope you will find this project useful and the program instructive. Then, let's see one of *your* PIC-oriented projects!

Bob Anding, AA5OY, has worked as a broadcast engineer for 21 years and is currently employed by WVUE-TV in New Orleans, LA. He attended the University of New Orleans from 1978 to 1981 pursuing studies in electrical engineering and computer science. Bob was first licensed in 1987 as N5LJR. In 1989, he upgraded to Extra class and changed his call sign to AA5OY. You can contact Bob at 1101 High Ave, Metairie, LA. 70001, e-mail **BANDING@worldnet.att.net**.

Further Reading

Easy Pic'n

A Beginners' Guide to PIC 16/17 Microcontrollers

By David Brenson, Square 1 Electronics, PO Box 501, Kelseyville, CA, tel: 707-279-8881; e-mail **squareone@zapcom.net**

ISBN 0-9654162-0-8

Easy Pic'n is a great book for the beginning PIC programmer. It assumes you have no prior knowledge of the subject and takes you through the first steps of PIC programming.

Pic'n up the Pace

PIC 16/17 Microcontroller Applications Guide

By David Brenson, Square 1 Electronics, PO Box 501, Kelseyville, CA, tel 707-279-8881; e-mail **sqone@pacific.net**; **http://www.sq-1.com**

ISBN 0-9654162-1-6

Pic'n up the Pace is the second book in the series. It picks up where the first, Easy PIC'n, leaves off. This book introduces you to basic applications of the PIC chip.

Embedded Control Handbook Vol 1 and Vol 2

Published Microchip Technology, Inc.

This book is available from retail suppliers and can also be downloaded from Microchip Web site (**http://www.microchip.com**) and the Microchip BBS (see the sidebar "**Programming Your Own PIC**"). It's a great information resource for the 16/17*xx* microcontrollers and a "must-have" book for anyone who programs PICs.

Programming Your Own PIC

If you want to program your own PIC, you will need a programmer and an IBM PC-compatible computer to load the program into the chip. Fortunately, the programmer schematic and software are readily available on the Internet—for *free*. You'll also find several schematics for programmers you can build yourself. One popular PIC programmer is designed by David Tait, G0JVY. [*****] It's very small and easy to assemble. David's programmer connects to a PC's parallel port. The software included with the package runs the programmer.

There are a number of other do-it-yourself PIC programmers on the Internet; some interface to the PC via its parallel port; others use the serial port. Still other PIC programmers are built for platforms other than PC compatibles. I spent a little time searching the Internet for PIC sites. I had an enjoyable time examining the different designs, and I picked up a lot of good information in the

process. Try it yourself.

Microchip's Web site is a good starting point (**http://www.microchip.com**). At this writing, Microchip's "Microchip Enthusiast Sites" offers links to different PIC sites on the Internet. While you are at Microchip's site, download their MPASM assembler. This free software compiles the PIC assembly code into hexadecimal (hex) code needed by the PIC ICs. You then use your programmer to load the hex code into the 16C84 chip. Microchip sells an assembled 16C84 programmer. This DOS-based programmer (Picstart-Lite) costs only $69 and is available from Digi-Key. The Picstart-Lite does *not* program the newer 16F84 chip, however. To program a 16F84 PIC, you need to use Microchip's Picstart-Plus programmer. Microchip files are available at their Web site and via their telephone BBS. [**]

Once you've built (or bought) your programmer, you need to test it. Most programmers have a test program to run in your PIC chip. With a successful test, you are ready to program your 16C84 chip. If you're entering the code by hand, you can use any *ASCII* text editor to save the resultant file to disk. (It's important that the editor saves an *ASCII* text file. Some word-processing programs include nonASCII characters with the text. Such nonASCII characters will cause the assembler to crash.) I use the DOS editor *EDIT*. *EDIT* is bundled free with DOS versions 5.0, and later.

Enter the code from the IDer's assembly language program listing and save the file as *IDER.ASM* in the same directory as the MPASM assembler. Follow the assembler's instructions to compile a hex file. When the assembler has compiled the program, it writes a file with the same name, but a different extension, *IDER.HEX*. Next, you load the hex file into the PIC chip using the programmer, following the programmer's instructions. *—Bob Anding, AA5OY*

Morse Code Encoding Scheme

Here's the PIC IDer's software Morse code character-encoding scheme. The program decodes the binary code and sends Morse code tones using 1s and 0s. When the program detects the last 1 in the string, it stops sending and gets the next letter.

Character	*Morse Code*	*Binary Encoding*	*Hex Encoding*
A	• —	00000110	06
B	— • • •	00010001	11
C	— • — •	00010101	15
D	— • •	00001001	09
E	•	00000010	02
F	• • — •	00010100	14
G	— — •	00001011	0b
H	• • • •	00010000	10
I	• •	00000100	04
J	• — — —	00011110	1e
K	— • —	00001101	0d
L	• — • •	00010010	12
M	— —	00000111	07
N	— •	00000101	05
O	— — —	00001111	0f
P	• — — •	00010110	16
Q	— — • —	00011011	1b
R	• — •	00001010	0a
S	• • •	00001000	08
T	—	00000011	03
U	• • —	00001100	0c
V	• • • —	00011000	18
W	• —	00001110	0e
X	— • • —	00011001	19
Y	— • — —	00011101	1d
Z	— — • •	00010011	13
0	— — — — —	00111111	3f
1	• — — — —	00111110	3e
2	• • — — —	00111100	3c

3	• • • — —	00111000	38
4	• • • • —	00110000	30
5	• • • • •	00100000	20
6	– • • • •	00100001	21
7	— — • • •	00100011	23
8	— — — • •	00100111	27
9	— — — — •	00101111	2f
/	— • • — •	00101001	29
SK	• • • — • —	01101000	68
AR	• — • — •	00101010	2a
BT	— • • • —	00110001	31
Word space	(WS)	00000000	00
End of Message	(EOM)	11111111	ff