

# Really PIC Key, PIC Key

*Add memory functions to your PIC keyer project.*

*Amateur radio experimenting is a fascinating and ongoing process, and we amateurs are driven by the sheer delight of learning by doing. And it is a long-standing convention that most amateur radio projects are in a state of continuous modification — the more we learn, the more we desire to incorporate into our projects.*

I've prepared this follow-up article in accordance with this tradition. My original PIC keyer project appeared in the September 1999 issue of *73 Amateur Radio Today*, and now that you've built the original circuit, it's time to enhance its performance! Let's begin by teaching the little hummer to automatically send frequently used CW messages.

Table 1 lists some sample messages, but of course you will use your own

personal data. Consider how great it will be to send the entire message with the single press of a button (and send perfect, machine-formed characters in the process!). Push the button, lean back, and wait for an answer. Ol' Morse and Marconi are probably looking down and smiling.

## Operational algorithm

The new operational algorithm is more complicated than the original one, but it takes a little patience and coaxing with code to entice the little PIC to perform new tricks.

Let's begin with Fig. 1. Notice the left bottom corner, the portion entitled "Is RA2 low?" After the words "Paddle control" and down to the end

is the same algorithm as the previous article (refer to Fig. 1 in *PIC Key, PIC Key*). We will only be discussing the new section of the algorithm — from "Start" to "Is RA2 low?" Let us make an excursion through the chart.

As usual, the program runs from the point labeled "Start". If you remember, the microcontroller PIC16F84 has 5 input/output lines at port A and 8 at port B. In previous programs, we programmed port A as input and all of port B as output. But actually we used only two lines as outputs — one for keying the transmitter and the other for audio control.

In this version of the program, it seems wiser to set all lines of port A, as well as almost all at port B, as inputs.

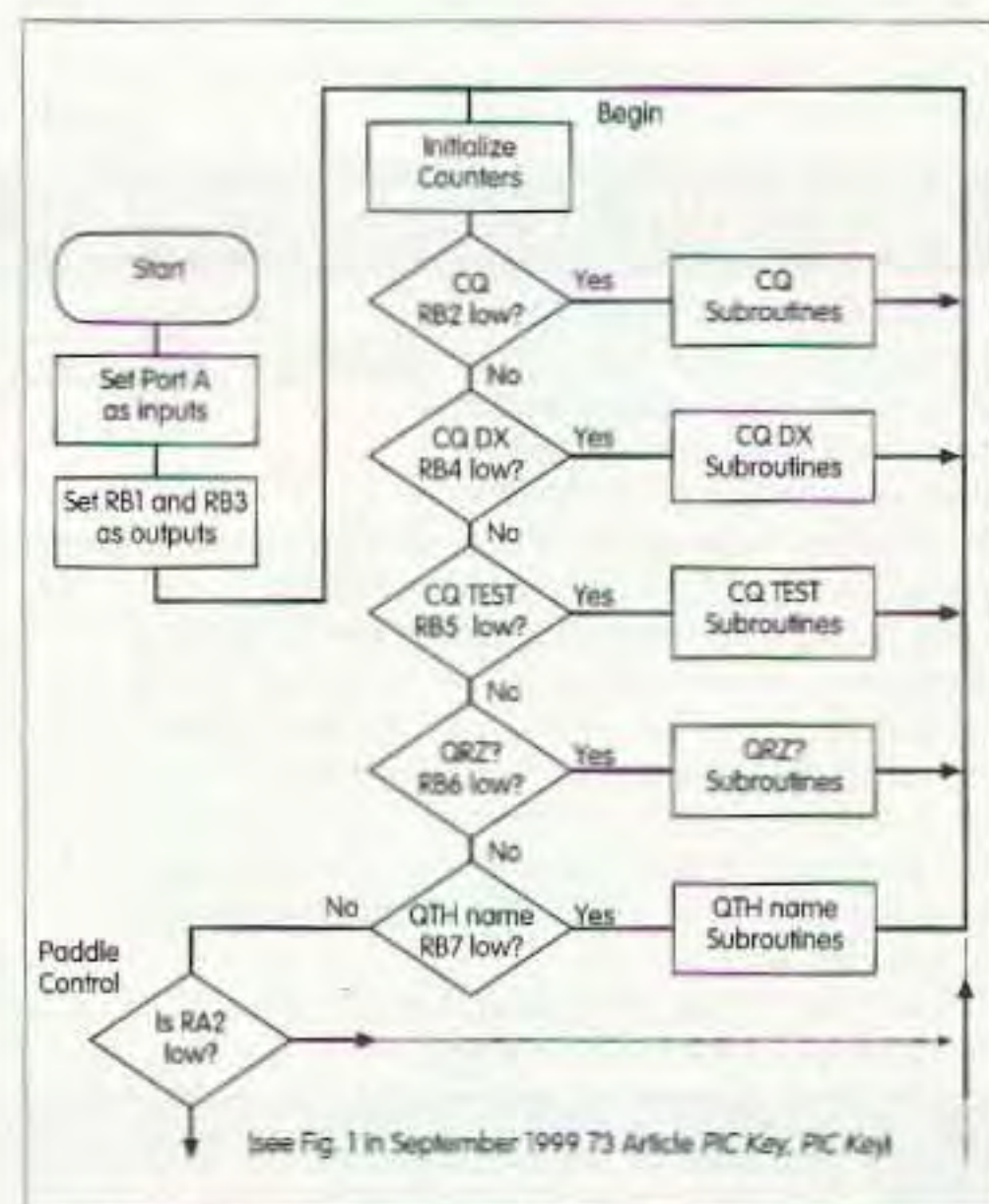


Fig. 1. Operation algorithm for the upgraded PIC-controlled keyer.

Message number	Message text
1	CQ CQ CQ de UY5DJ UY5DJ UY5DJ PSE K
2	CQ DX CQ DX CQ DX de UY5DJ UY5DJ UY5DJ PSE K
3	CQ TEST CQ TEST de UY5DJ UY5DJ TEST K
4	QRZ? QRZ? de UY5DJ UY5DJ PSE K
5	My QTH is Kharkiv Kharkiv es name is Vlad Vlad PSE K

Table 1. Frequently used CW messages.

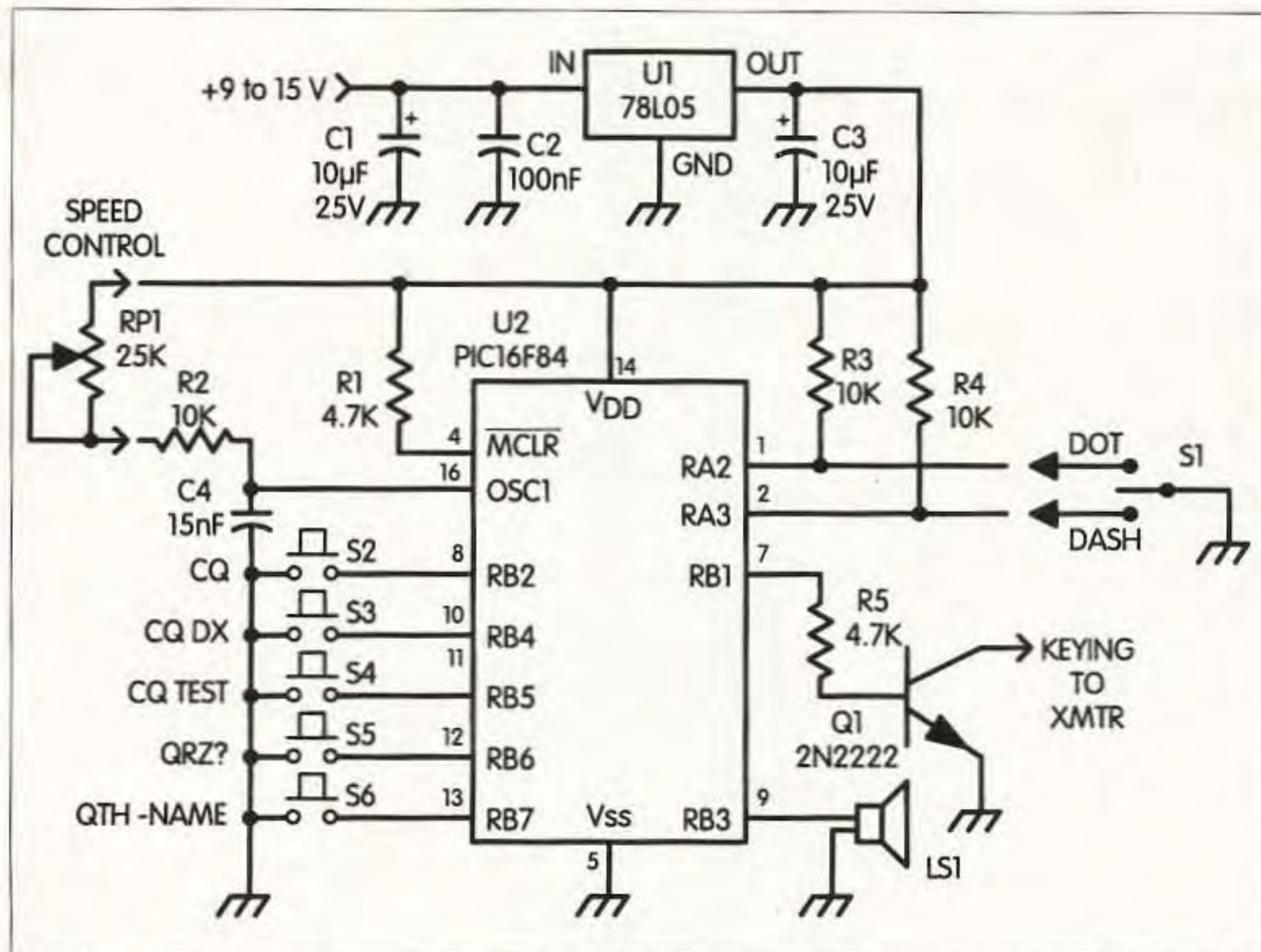


Fig. 2. Schematic of the improved keyer.

There are two exceptions — RB1 and RB3 are left as outputs. The next step is to initialize all counters in the program. This point is rather important, and marked by the label "Begin". Many times during operation, the program will come here and begin its run down to the end.

As you sensed while reading above, this keyer has several new input push-buttons connected to the appropriate inputs of the microcontroller. By pressing one button we make that input low, which creates the desired message.

After initialization, the program checks to see if a pressed button is connected to the RB2 line. If it was pressed, the CQ message (number 1 in Table 1) is requested. The program

will go to the "yes" direction to run the set of CQ subroutines. It causes transfer of the Morse code signals to the output. When the message is completed, the program goes back to the "Begin" label and everything will repeat.

When RB2 isn't low, the program will check to see if the "CQDX" button was pressed. It can find RB4 either high or low. If it is low, the program sends message number 2 from the table. In the opposite case, the program checks for low condition and consequently port lines RB5, RB6, and RB7. If it finds any low, the program sends the appropriate message and returns to the re-initialization of the counters. If no buttons were pressed, the program, after the last examination of RB7, continues

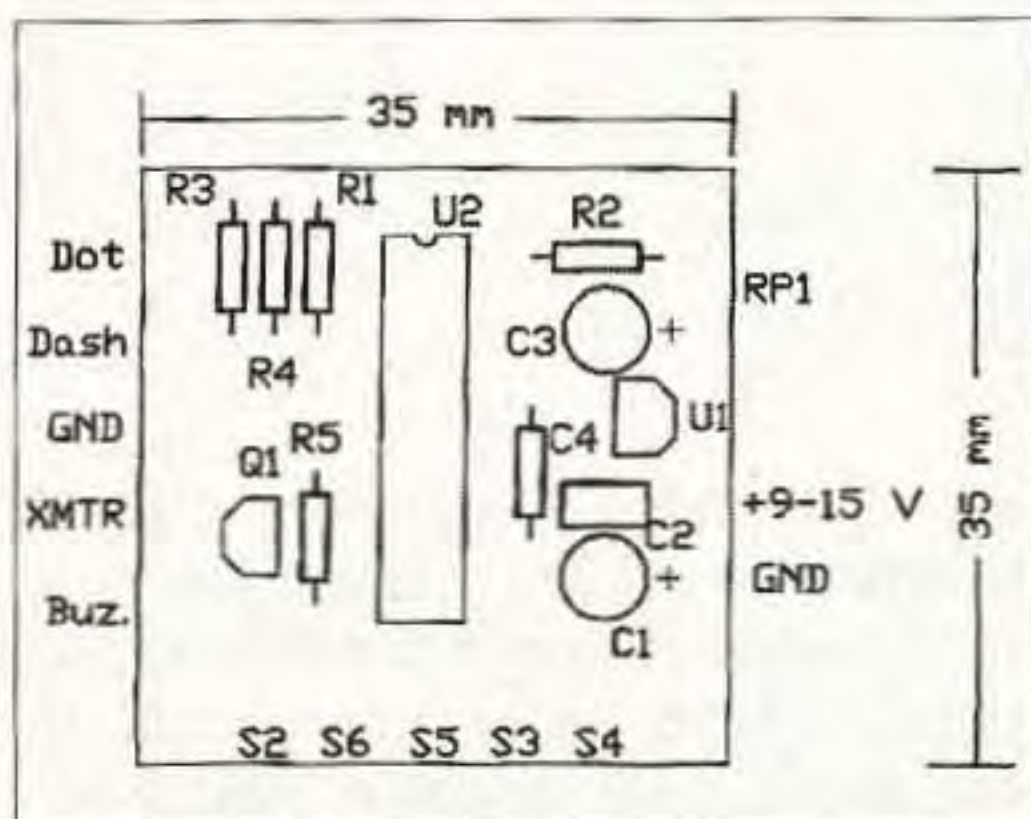


Fig. 3(a). PIC keyer PC board, component side.

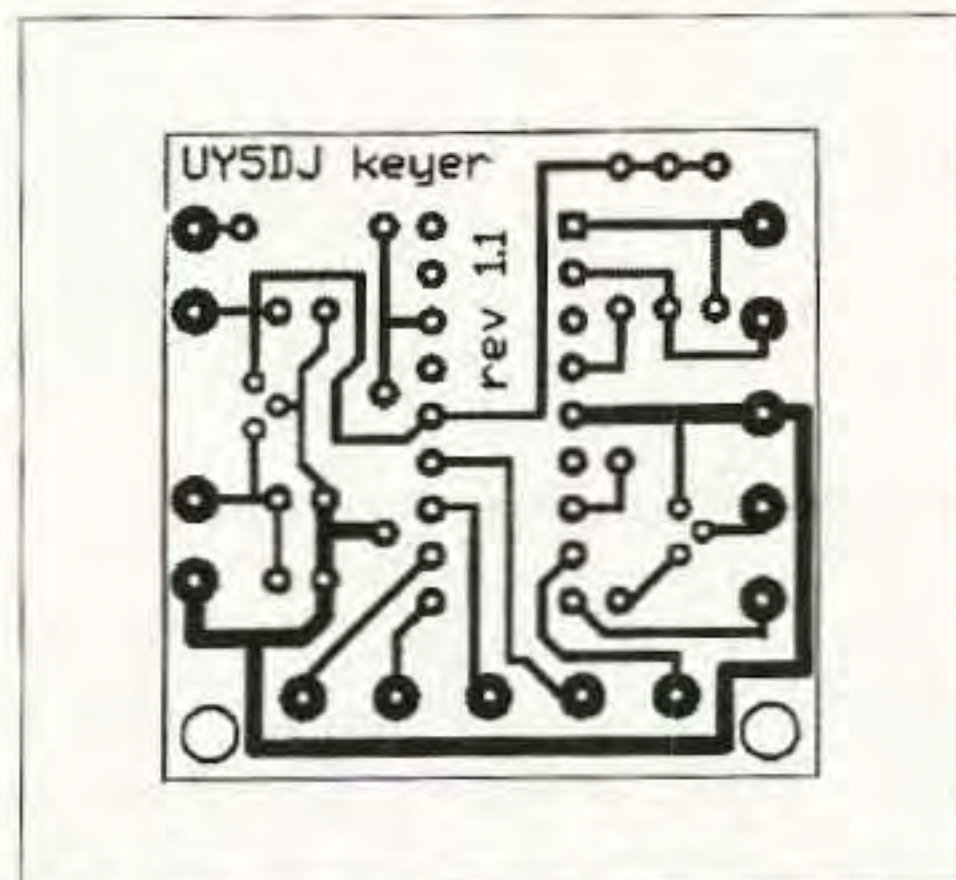


Fig. 3(b). PIC keyer PC board, foil side.

# ALL ELECTRONICS CORPORATION

C O R P O R A T I O N

## Laser Level



Use it to match heights in large rooms or across buildings. Locking push button switch prevents unintended actuation. Includes two AAA batteries. **\$16<sup>95</sup>** each

CAT # LL-1

## 16 Character X 2 Line LCD with Backlight

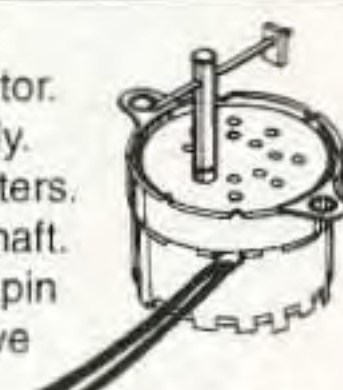
Daewoo # 16216L-5-VSO  
5 x 7 dot format.  
2.56" x 0.54" viewing area.  
3.15" x 1.41" module size.  
LED backlight. Includes hook-up/spec sheet.



CAT# LCD-53 **\$7<sup>50</sup>** each

## 40 RPM 115 Vac Motor

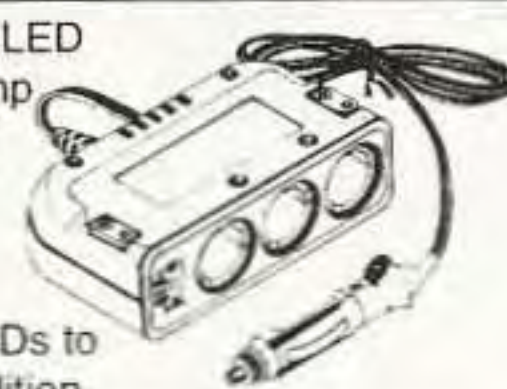
Cramer Co # 105VA150-E  
40 RPM, 115 Vac gear motor.  
1.83" diameter x 1.18" body.  
Mounting ears on 2.1" centers.  
0.25" diameter x 1" long shaft.  
Prepped with a 1.25" long pin through the hole in the drive shaft. Pin is removable.  
4" long pigtail leads. UL, CSA.



CAT # ACM-108 **\$2<sup>50</sup>** each

## 3 Outlet Lighter Cord w/ Battery Monitor

Three foot cord with LED lighted, fused, 10 Amp plug at one end and three outlet jacks at other end. Jack assembly has red, yellow and green LEDs to indicate battery condition. Can be mounted via mounting ears (4.7" centers) or double-sided tape (included). Mounting ears fold out of way if not in use.



CAT # CLP-44 **\$3<sup>75</sup>** each

ORDER TOLL FREE  
**1-800-826-5432**  
SHOP OUR ONLINE STORE  
[www.allelectronics.com](http://www.allelectronics.com)  
CHARGE ORDERS to Visa, Mastercard, American Express or Discover

TERMS: NO MINIMUM ORDER. Shipping and handling for the 48 continental U.S.A. \$5.00 per order. All others including AK, HI, PR or Canada must pay full shipping. All orders delivered in CALIFORNIA must include local state sales tax. Quantities Limited. NO COD. Prices subject to change without notice.

CALL, WRITE FAX or E-MAIL for our FREE 96 Page CATALOG Outside the U.S.A. send \$3.00 postage.

MAIL ORDERS TO:  
**ALL ELECTRONICS CORPORATION**  
P.O. Box 567  
Van Nuys, CA 91408  
FAX (818)781-2653

e-mail [allcorp@allcorp.com](mailto:allcorp@allcorp.com)

```

May 1999
CW keyer
ver. 1.1
Vlad Skrypnik, UT5DJ      E-mail: uy5dj@yahoo.com
-----
list      p=14F54
config    0a3ff3      : RC clock oscillator
-----
CPU equates (memory map)
porta     equ     0x05
portb     equ     0x06
count1    equ     0x0c      :for DOT delay constant
count2    equ     0x0d      :for PAUSE delay constant
count3    equ     0x0e      :for DASH delay constant
count4    equ     0x0f      :for 2xPAUSE delay constant
count5    equ     0x10      :for 5xPAUSE delay constant
-----
start     org     0x000
          movlw  0x00
          tris   porta      ; teach port A as inputs
          movlw  0xf3
          tris   portb      ; teach port RB1 and RB2 as outputs
          bcf   portb,1      ; set RB1 low
          bcf   portb,3      ; set RB3 low
          movlw  0x7f      ; \ set internal pullup resistors
          option ;/ on port B inputs enabled
-----
begin     cldc   count1      ; initialize counters
          cldc   count2
          cldc   count3
          cldc   count4
          cldc   count5
-----
          Calling CQ
MainCQ    btfsc  portb,2      ; is RB2 low? (pin 8)
          goto  CQOK
          call  CQ
          call  CQ
          call  CQ
          call  from
          call  MyCall
          call  MyCall
          call  MyCall
          call  PSE
          call  K
          goto  begin
-----
          Calling CQ DX
CQOK      btfsc  portb,4      ; is RB4 low? (pin 10)
          goto  CQTEST
          call  CQ
          call  DX
          call  CQ
          call  DX
          call  from
          call  MyCall
          call  MyCall
          call  MyCall
          call  PSE
          call  DX
          call  K
          goto  begin
-----
          Calling CQ TEST
CQTEST    btfsc  portb,5      ; is RB5 low? (pin 11)
          goto  QRZ
          call  CQ
          call  TEST
          call  CQ
          call  TEST
          call  from
          call  MyCall
          call  MyCall
          call  TEST
          call  K
          goto  begin
-----
          Calling QRZ?
QRZ       btfsc  portb,6      ; is RB6 low? (pin 12)
          goto  QTHname
          call  QRZ?
          call  QRZ?
          call  from
          call  MyCall
          call  MyCall
          call  PSE
          call  K
          goto  begin
-----
          Send QTH and name
QTHname   btfsc  portb,7      ; is RB7 low? (pin 13)
          goto  paddle
          call  dash      ; letter "n"
          call  dash
          call  pause2
          call  dash      ; letter "y"
          call  dot
          call  dash
          call  dash
          call  LP
          call  dash      ; letter "Q"
          call  dash
          call  dot
          call  dash
          call  pause2
          call  dash      ; letter "T"
          call  pause2
          call  dot      ; letter "H"
          call  dot
          call  dot
          call  dot
          call  LP
          call  is
          call  MyQTH
          call  MyQTH
          call  dot      ; letter "e"
          call  pause2
          call  dot      ; letter "s"
          call  dot
          call  LP

```

```

          call  dash      ; letter "n"
          call  dot
          call  pause2
          call  dot      ; letter "e"
          call  dash
          call  pause2
          call  dash      ; letter "m"
          call  dash
          call  pause2
          call  dot      ; letter "o"
          call  LP
          call  is
          call  MyName
          call  MyName
          call  PSE
          call  K
          goto  begin
-----
          Subroutine "CQ"
CQ        call  dash
          call  dot
          call  dash
          call  dot
          call  pause2
          call  dash
          call  dash
          call  dot
          call  dash
          call  LP
          return
-----
          Subroutine "DE"
from      call  dash
          call  dot
          call  dot
          call  pause2
          call  dot
          call  LP
          return
-----
          Subroutine "DX"
DX        call  dash
          call  dot
          call  dot
          call  pause2
          call  dash
          call  dot
          call  dot
          call  dash
          call  LP
          return
-----
          Subroutine "TEST"
TEST      call  dash
          call  pause2
          call  dot
          call  pause2
          call  dot
          call  dot
          call  dot
          call  pause2
          call  dash
          call  LP
          return
-----
          Subroutine "IS"
is        call  dot      ; letter "i"
          call  dot
          call  pause2
          call  dot      ; letter "s"
          call  dot
          call  LP
          return
-----
          Subroutine "CALL SIGN"
MyCall    call  dot      ; letter "w"
          call  dot
          call  dash
          call  pause2
          call  dash      ; letter "y"
          call  dot
          call  dash
          call  dash
          call  dash
          call  dash
          call  dot
          call  dash
          call  dot
          call  pause2
          call  dash      ; letter "d"
          call  dot
          call  dot
          call  pause2
          call  dot      ; letter "j"
          call  dash
          call  dash
          call  LP
          return
-----
          Subroutine "My QTH"
MyQTH     call  dash      ; letter "k"
          call  dot
          call  dash
          call  pause2
          call  dot      ; letter "h"
          call  dot
          call  dot
          call  dot
          call  pause2
          call  dot      ; letter "a"
          call  dash
          call  pause2
          call  dot      ; letter "r"
          call  dash
          call  dot
          call  pause2
          call  dot      ; letter "x"
          call  dash
          call  dot
          call  pause2
          call  dot      ; letter "i"
          call  dot
          call  pause2
          call  dot      ; letter "v"
          call  dot

```

to the paddle control. You can manually manipulate the paddle to send either a dash or dot to the output. To recall how it works, please refer to the algorithm chart in the previous article. If no dash or dot inputs were low, the program comes back to "Begin".

### Assembly language program

The structure of this assembly program was detailed in the previous article. **Table 2** shows that there are some differences even at the CPU equates. There are two more counters

added. Counter 4 will keep a delay constant for pauses between letters in the message. Counter 5 will store the delay constant to separate words. In the Morse code structure, pauses between letters are three times longer than pauses between dots and dashes

```

call dot
call dash
call LP
return
----- Subroutine "My name" -----
MyName call dot ; letter "v"
call dot
call dash
call pause2
call dot ; letter "l"
call dash
call dot
call dot
call pause2
call dot ; letter "e"
call dash
call pause2
call dash ; letter "d"
call dot
call dot
call LP
return
----- Subroutine "QRZ?" -----
QRZ? call dash
call dash
call dot
call dash
call pause2
call dot
call dash
call dot
call pause2
call dash
call dot
call dot
call pause2
call dot
call dot
call dash
call dash
call dot
call dot
call dot
call LP
return
----- Subroutine "PSE" -----
PSE call dot
call dash
call dash
call dot
call pause2
call dot
call dot
call dot
call pause2
call dot
call LP
return
----- Subroutine "X" -----
X call dash
call dot
call dash
return
----- Subroutine of pauses between letters -----
pause2 movlw d'24' ; delay constant
movwf count4 ; load counter with delay const
rptpa2 decfz count4,f ; decrement counter
goto rtpa2 ; not 0
return ; counter 0, end pause
----- Subroutine of pauses between words -----
LP movlw d'102' ; delay constant
movwf count5 ; load counter with delay const
rtpa5 decfz count5,f ; decrement counter
goto rtpa5 ; not 0
return ; counter 0, end pause
----- Manipulating by paddle -----
----- Select dot -----
paddle btfsc porta,2 ; is RA2 low (dot pressed)?
goto dash?
call dot ; calling subroutine DOT
goto begin
----- Select dash -----
dash? btfsc porta,3 ; is RA3 low (dash pressed)?
goto begin
call dash ; calling subroutine DASH
goto begin
----- Subroutine for generating dots -----
dot bcf portb,1 ; RB1=1, dot begins
movlw d'12' ; delay constant
movwf count1 ; load const to counter
rptdot bcf portb,3 ; sound on
bcf portb,3 ; sound off
decfz count1,f ; decrement counter
goto rptdot ; not 0
bcf portb,1 ; RB1=0, end dot
call pause ; start PAUSE subroutine
return
----- Subroutine for generating dashes -----
dash bcf portb,1 ; RB1=1, dash begins
movlw d'37' ; delay constant
movwf count3 ; load const to counter
rptdash bcf portb,3 ; sound on
bcf portb,3 ; sound off
decfz count3,f ; decrement counter
goto rptdash ; not 0
bcf portb,1 ; RB1=0, end dash
call pause ; start PAUSE subroutine
return
----- Subroutine for generating pauses between elements -----
pause movlw d'9' ; delay constant
movwf count2 ; load counter with delay const
rtpau decfz count2,f ; decrement counter
goto rtpau ; not 0
return ; counter 0, end pause
----- END of program -----
end

```

in the letter. It is equal to the length of a single dash. The duration of the pause between words is equal to three dashes or nine dots.

After you are familiar with assembly programming, you can easily understand what has happened in the lines preceded by labels "Start" and "Begin". After initialization of counters (merely clearing their memory cells), the program

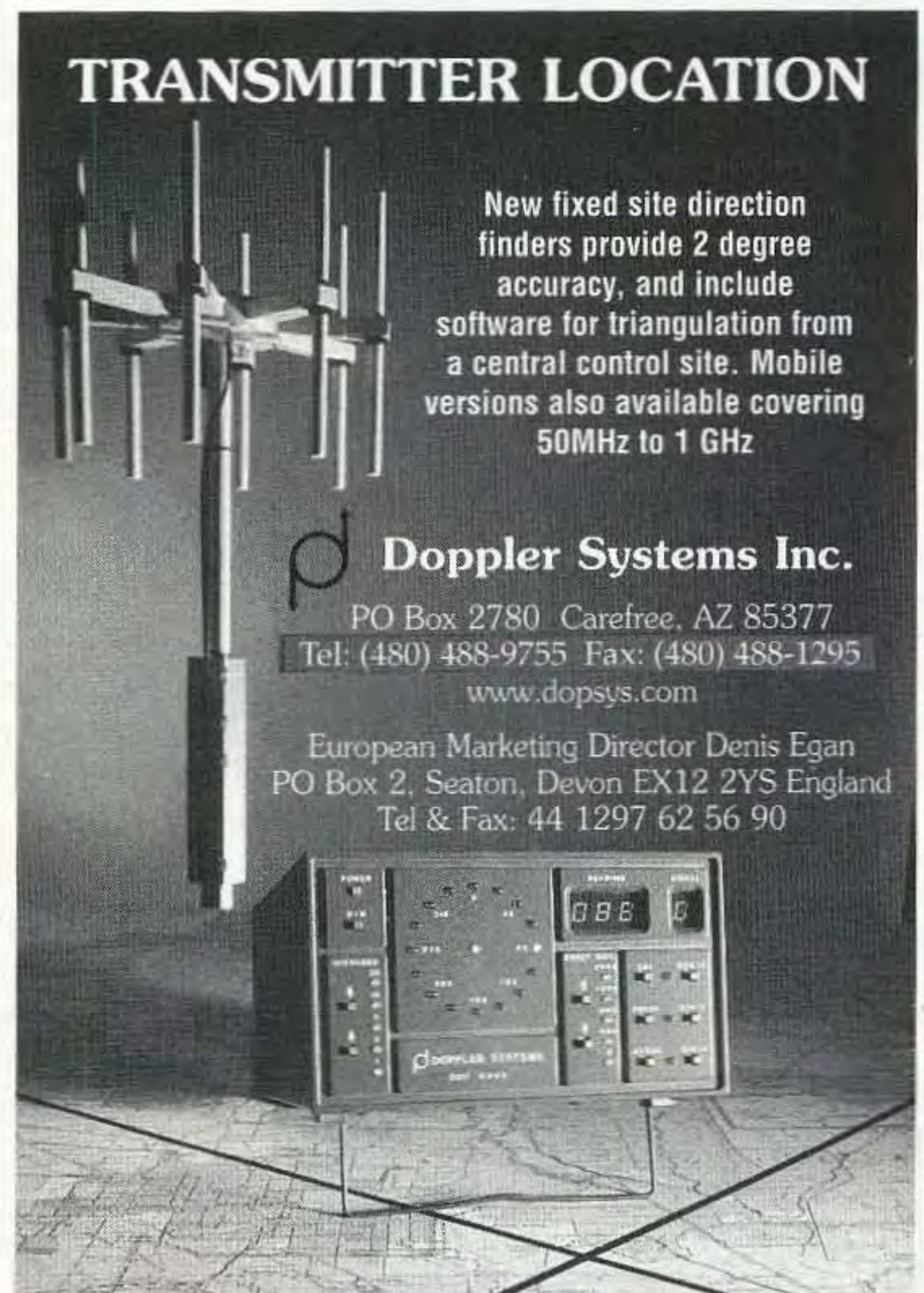
Table 2. An assembly language program for a modified PIC keyer.

Name	Description
C1, C3	10 $\mu$ F 25 V electrolytic or tantalum (DK P5148-ND)
C2	100 nF ceramic (DK P4924-ND)
C4	15 nF ceramic (DK P4905-ND)
LS1	Piezo buzzer element (DK P9924-ND)
Q1	2N2222 or any general purpose NPN silicon transistor (DK PN2222ADICT-ND)
RP1	25k potentiometer (DK CT2266-ND)
S1	Any type CW keyer paddle
S2-S6	Any type push-button switches (e.g., DK P8006S-ND)
U1	78L05 small 5 V positive regulator (DK NJM78L05A-ND)
U2	PIC16F84 microcontroller (DK PIC16F84-04/P-ND)

Table 3. Parts list.

**A GREAT gift idea for yourself, your ham friend(s), or your child's school library**  
 is a subscription to 73 Magazine ... only \$24.97!  
 Call 800-274-7373 or write to 70 Route 202 North,  
 Peterborough NH 03458

## TRANSMITTER LOCATION



New fixed site direction finders provide 2 degree accuracy, and include software for triangulation from a central control site. Mobile versions also available covering 50MHz to 1 GHz

**Doppler Systems Inc.**  
 PO Box 2780 Carefree, AZ 85377  
 Tel: (480) 488-9755 Fax: (480) 488-1295  
 www.dopsys.com

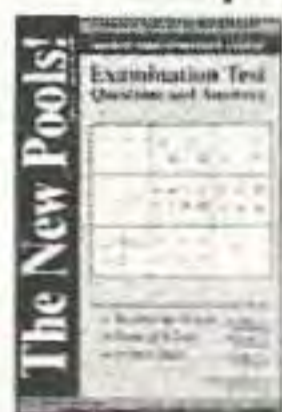
European Marketing Director Denis Egan  
 PO Box 2, Seaton, Devon EX12 2YS England  
 Tel & Fax: 44 1297 62 56 90

**Back Issues**  
of  
**73 Magazine**  
Only \$5.00 Each!  
Call 800-274-7373

## The New Pools!

Examination Test  
Questions & Answers

*The exact questions, multiple choices  
and answers for the Technician Class,  
General Class, and Extra Class  
operator's license.*



Only  
\$9.95  
Plus \$3.50 S&H

**Omega Sales**

P.O. Box 376  
Jaffrey, NH 03452  
800-467-7237



## From MILLIWATTS to KILOWATTS™

### RF PARTS HAS IT!

Complete inventory for servicing  
Amateur, Marine, and Commercial  
Communications Equipment.

- Transmitting Tubes & Sockets
- RF Power Transistors
- VHF/UHF RF Power Modules
- Low Noise RF FET's
- Bird Electronics Wattmeters
- Doorknob Capacitors
- Chokes • Broadband Transformers

Se Habla Español • We Export

Visit our Web Site for latest  
Catalog pricing and Specials:  
<http://www.rfparts.com>

ORDERS ONLY  
1-800-RF-PARTS • 1-800-737-2787

ORDER LINE • TECH HELP • DELIVERY INFO.  
760-744-0700

FAX TOLL-FREE FAX  
760-744-1943 888-744-1943

E-MAIL: [rpf@rfparts.com](mailto:rpf@rfparts.com)



**RF PARTS**  
435 SOUTH PACIFIC STREET  
SAN MARCOS, CA 92069

starts to examine five port B inputs. The first step is labeled "MainCQ". If RB2 is still high (button unpressed) the next line instruction, "Goto CQDX", skips over the CQ message and examines input RB4.

If RB2 was really low, the program will ignore the second line and continue from the third one. Actually, there are numerous subroutines being called one by one while forming the message. At first, subroutine CQ was called three times. Next is called subroutine "from", which in fact generates the word "de" in the radio message. (It is impossible to name this subroutine as "de", because this combination of letters is reserved for the PIC microcontroller and is forbidden for use as the label or subroutine name.)

Then subroutine "MyCall" was called three times, "PSE" once, and "K". The first CW message is completed, and the instruction "Goto" returns the program to "Begin."

I hope this gives you the idea of how any message is formed. You may examine how it is organized in the "CQDX", "CQTEST", "QRZ," and "QTHname" portions of this program. It is really very easy. Subroutines included here also invoke other subroutines for dots, dashes, and pauses.

Please pay careful attention to subroutines "MyCall," "MyQTH", and "Name." You must understand how to change the sets of dots, dashes, and pauses to make your callsign, QTH, and name available. Remember that you have to call each time the subroutine produces one dot, dash, or pause. These subroutines are only what you need to change for correct operation of this keyer at your station. First, merely write your callsign, QTH, and name in Morse code, using dots and dashes. Then substitute them by instruction "Call" and appropriate subroutine name. Please keep in mind that pauses between dots or dashes are included into both subroutines generating dots and dashes (at their end). This means that you do not add any pause after Morse code elements.

Subroutines "Pause2" and "LP" provide pauses between letters and words in the messages. The required duration of these pauses was achieved by appropriate selection of the delay constant's values.

The part of the program labeled "paddle" is almost the same as what was in the original keyer program. One difference is in subroutine "Pause." The delay constant was changed from 14 to 9. Why? Because when the keyer is operating from the paddle any time the program is checking five microcontrollers' inputs and uses 5 processor cycles more each time. This, of course, will increase the pause duration between dots and dashes. To compensate for this, the delay constant was decreased.

### Schematic diagram and construction

Fig. 2 shows the schematic diagram of the improved PIC keyer. The only differences from the original keyer are push-buttons S2-S6. They are normally open, and are intended to pull the PIC's inputs to ground. This will activate one of the previously determined messages.

The new keyer is assembled on a small 35 x 35 mm single-sided printed circuit board (Fig. 3). Please note that pads are provided for connecting the push-buttons. It will give you a variety of choices in your selection of push-buttons, paddle, and cabinet for final construction.

### Summary

Building this simple keyer will help you gain knowledge and skills through study, experimentation, and construction — and you will end up with a very useful station accessory as well! Like most amateur radio projects, this project is ripe for further improvements and modifications. Keep in mind that the program described in this article utilizes only a very small part of the PIC16F84's capabilities.

I want to express my gratitude to my friend Dave Evison W7DE for his patience in reading and doing some preliminary editing of this article. 