A MORSE TUTOR AND MEMORY KEYER USING THE SINCLAIR ZX81 COMPUTER

J. V. MOSS, B.SC., A.M.B.C.S., G4ILO

ONE of the biggest problems facing the aspiring Class-A licensee is simply getting enough code practice, at the right speed and at a convenient time. The author, having given up with records and tapes, eventually reached the vital 12 w.p.m. after programming a Nascom 1 microcomputer to provide him with an endless supply of random Morse.

There are now commercial Morse tutors on the market, and they cost much less than did the Nascom 1. The Sinclair ZX81, on the other hand, costs about the same as a Morse tutor, and as well as being programmed to become one, it can do many other things. This article describes how to get the ZX81 to send Morse, and how to use this to make a Morse tutor, a Morse keyboard and even a memory keyer for meteor-scatter work. All the programs will run on the basic machine without additional memory.

Morse Code Routines

The ZX81 has one output port, the one used for saving programs to cassette. A tone is generated by setting the port to 1, then 0, very rapidly, giving a square wave; the duration of the tone gives the Morse dots and dashes. Unfortunately, it is not possible to set and unset the port using BASIC, and so the programming to generate the Morse characters must be done in machine code[1].

A listing of the machine code routines is given at Table 5. However, it is not necessary to understand the routines in order to use them, and they may be loaded into the ZX81 by the following procedure:

(*a*) Clear the machine by typing NEW, then input a line 1 REM followed by 250 A's.

PRINT PEEK 16511 will give a value 2 more than the number of A's in the line. There must be 250 (*i.e.* PEEK 16511 = 252).

(*b*) Now enter the BASIC program of Table 1A.

(*c*) RUN the program. It will display 16514. Enter the first number of the data in Table 1B: 118. The program will then display 16515. Enter the next number working along the row, 118 again. What you are doing is storing the machine code program in memory locations 16514 to 16763.

Table 1A: Machine Code Input Program

```
 1 REM AAA . . . (250 letter A's)
10 LET L = 16514
20 LET S = 0
25 CLS
30 FOR I = L TO L + 9
40 PRINT I,
50 INPUT X
60 POKE I, X
70 PRINT X
80 LET S = S + X
90 NEXT I
100 PRINT "CHECK", S
110 PRINT "OK?"
120 INPUT A$
130 IF A$ = "Y" THEN LET L = L + 10
140 GOTO 20
```

Table 1B: Machine Code Data

Location	*Values*										*Check*
16514	118	118	80	0	0	0	42	132	64	62	616
16524	4	211	255	6	128	16	254	205	67	15	1161
16534	6	128	16	254	45	32	238	201	42	132	1094
16544	64	6	0	16	254	45	32	249	201	58	925
16554	133	64	198	4	24	5	58	133	64	198	881
16564	2	205	158	64	61	32	250	201	33	134	1140
16574	64	86	30	4	203	2	203	2	62	3	659
16584	162	40	16	33	222	64	133	111	14	1	796
16594	233	29	32	236	33	135	64	24	228	205	1219
16604	176	64	201	12	12	205	136	64	13	32	915
16614	250	205	158	64	24	231	237	75	16	65	1325
16624	121	254	11	56	19	33	252	64	9	9	828
16634	126	50	134	64	35	126	50	135	64	205	989
16644	188	64	24	3	205	169	64	237	75	16	1045
16654	65	201	0	0	223	112	0	0	0	0	601
16664	87	240	245	240	117	192	117	208	223	192	1861
16674	119	112	127	64	221	192	127	208	125	0	1295
16684	125	192	119	112	95	80	221	208	85	64	1301
16694	213	64	245	64	253	64	255	64	255	192	1669
16704	127	192	95	192	87	192	85	192	208	0	1370
16714	127	0	119	0	124	0	192	0	247	0	809
16724	92	0	255	0	240	0	213	0	116	0	916
16734	223	0	80	0	112	0	84	0	215	0	714
16744	93	0	220	0	252	0	64	0	244	0	873
16754	253	0	212	0	125	0	117	0	95	0	802

(*d*) When you have entered the ten values of the first row, the computer will display a check number. This should agree with the check number at the end of the row in Table 1B. If it does not, you have made a mistake typing in the numbers. Since the computer is displaying OK? answer it N (for No). You may then input the ten values of that row again.

(*e*) if the check number does agree, answer Y (for yes). You may then enter the next row of data. Continue this process until all the data of Table 1B has been entered.

(*f*) Delete the program of Table 1A (lines 10 to 140) leaving only line 1 REM. This line cannot now be listed, as the 250 A's have been replaced by the machine code routines. To list any BASIC lines following the REM, you will have to type LIST 2.

(*g*) SAVE the machine code REM line on tape.

The most difficult part of the job is now over, and the machine code routine may now be tested. The entry point of the main routine is 16620, and it expects 0 (for space) or a character code to be 'poked' into location 16656. Location 16516 is set to 800 ÷ speed in w.p.m., and is currently set to give a speed of 10 w.p.m. Location 16517 is used to give an extra delay between characters.

Since the ZX81 cannot produce the TV display and output to the cassette port at the same time, the computer must be set to FAST mode.

Typing in POKE 16656, CODE "K" followed by RAND USR 16620 will result in a pattern on the screen as the computer sends "K". If the cassette recorder is set to record from the output port, then the K may be recorded and played back. Unfortunately, the output level from the port is too low to drive even an earpiece directly, and so some amplification will be necessary in order to drive headphones or a loudspeaker.

The machine code routines can be used to generate Morse for many applications. All the ZX81 characters with codes from 11 to 63 with the exception of £ and $ may be sent, that is the numbers, letters an punctuation. All the BASIC program has to do is to POKE into 16656 the code of the character to be sent, and then RAND USR 16620 or LET A = USR 16620. If it is required to print the character to the screen as well, then PRINT USR 16620 could be used instead.

Morse Tutor

Table 2 gives a BASIC program to generate random five-character groups of letters only, numbers only, or mixed letters and numbers. The program asks the speed required in WPM, then

Table 2: Morse Tutor Program

```
 1 REM machine-code routines.
10 PRINT "WPM?"
20 INPUT X
30 POKE 16516,800/X
40 PRINT "DELAY?"
50 INPUT X
60 POKE 16517,X
70 LET Y=28
80 LET LETTERS=26
90 LET NUMBERS=10
100 PRINT "TEST?"
110 INPUT X
120 IF X=26 THEN LET Y=38
180 CLS
200 FOR I=1 TO 5
220 LET C=INT ( RND * X ) + Y
230 POKE 16656,C
240 PRINT CHR$ USR 16620;
250 NEXT I
255 POKE 16656,0
260 PRINT CHR$ USR 16620;
290 GOTO 200
```

it asks if a delay is wanted. The delay is measured in dot-lengths at the specified speed; 0 would be no delay, 5 would amount to a space between every character. The program then asks TEST? There are three possible replies: LETTERS, NUMBERS or LETTERS + NUMBERS. The computer will then begin sending. When it has filled up the screen (less than a full screen on a 1K machine) it will stop and display the characters sent for checking. The program may then be resumed by typing CONT.

If it is desired to produce a continuous stream of Morse without displaying the characters to the screen, for example, to produce practice tapes, then lines 240 and 260 of the program may be changed to LET A = USR 16620.

Morse Keyboard

Table 3 gives a program to turn the ZX81 into a Morse keyboard with programmable memory. The program again asks the speed required in WPM, it then asks MEM? A message of up to about 100 characters may be entered on a 1K machine. The TV screen then goes blank as the computer is ready to send Morse. As each key is pressed, the corresponding character is sent. The keys are not buffered, and so it is necessary to hold down a key until the previous character has been sent. If the key $ is pressed, then the message in memory will be transmitted.

Table 3: Morse Keyboard Program

```
 1REM machine code routines
 10 PRINT "WPM?"
 20 INPUT W
 30 POKE 16516, 800/W
 40 PRINT "MEM?"
 50 INPUT A$
 90 CLS
100 LET W = CODE INKEY$
110 IF W > 63 THEN GOTO 100
115 IF W < CODE":" THEN GOTO 200
120 POKE 16656, W
130 LET W = USR 16620
140 GOTO 100
200 IF W < CODE"$" THEN GOTO 100
310 FOR I = 1 TO LEN A$
320 POKE 16656, CODE A$(I)
330 LET W = USR 16620
340 NEXT I
350 GOTO 100
```

Interface

Of course, the ZX81 cannot key a transmitter directly, and so a circuit would be required to make the tone output drive a relay. Such a circuit as is used by meteor-scatter operators to key a transmitter with speeded-up Morse from a tape recorder would be suitable. Alternatively the tone output could be fed, *via* a suitable low-pass filter, into the microphone input of an SSB transmitter.

Another possibility is the plug-in data port for the ZX81 made by *Technomatic Ltd.* an described in [2]. This add-on board allows a relay to be controlled by the computer, which could be used to key a transmitter in the normal way. Minor alterations to the machine-code routines would be required if this board were to be used, and this has not been tried by the author.

MS Keyer

The final program is for a memory keyer for meteor-scatter work (Table 4). The keyer has two memories, one to hold the transmiting station's callsign, and one to hold the message to be sent using high-speed CW. The program will send the callsign at 12 w.p.m., followed by the message at 50 w.p.m. for about 55 seconds, and then the callsign again at 12 w.p.m., giving a total 'over' of about one minute.

Table 4: MS Memory Keyer

```
 1 REM Machine-code routines
 10 PRINT "CALL?"
 20 INPUT A$
 30 LET C$ = "DE " + A$
 40 IF INKEY$ = "R" THEN GOTO 100
 50 IF INKEY$ <> "P" THEN GOTO 40
 60 PRINT "MEM?"
 70 INPUT A$
 80 CLS
100 POKE 16516, 64
110 LET X$ = C$
120 GOSUB 500
200 POKE 16516, 16
210 LET X$ = A$
220 FOR I = 1 TO 240 / LEN X$
230 GOSUB 500
240 NEXT I
300 POKE 16516, 64
310 LET X$ = C$
320 GOSUB 500
330 GOTO 40
500 FOR K = 1 TO LEN X$
510 POKE 16656, CODE X$(K)
520 RAND USR 16620
530 NEXT K
540 RETURN
```

When the program is run, it will ask CALL? and the station callsign should be typed in. The screen will then go blank, and the program is waiting for one of two commands, R or P. P will cause the screen to display MEM? and the message to be sent repeatedly at high speed should be programmed in. The "newline" at the end of the message should be typed at the start of the transmission period, as the computer will immediately begin to send the message, with the station identification at 12 w.p.m. at the beginning and end of the transmission. R will cause the message already in the memory to be tranmitted.

Line 220 in the program determines the length of transmission, and is set to give about a minute, although the exact time will depend on the text of the message. It is a simple matter to alter the program to give a longer or shorter transmission periods, or to vary the speed of the sending.

Conclusion

The example programs given here have shown that it is possible to program the ZX81 to create sophisticated memory keyers. ZX81 owners with the 16K RAM pack will have far greater possibilities, including message storage capacity far in excess of any memory keyer commercially available. It should not be difficult for the average amateur, using these programs as examples, to create a microcomputer Morse keyer tailored to his individual requirements.

Table 5: Listing of Machine Code Routines

```
16514  76              HLT
16515  76              HLT
16516  50        SPEED: 50
16517  00        DELAY: 00
16518  00 00     CWCH:  00 00
16520  2A 84 40  TONE:  LD HL,(SPEED)
16523  3E 04     TO:    LD A,04
16525  D3 FF            OUT (FF),A
16527  06 80            LD B,80
16529  10 FE     T1:    DJNZ T1
16531  CD 43 0F         CALL 0F43
16534  06 80            LD B,80
16536  10 FE     T2:    DJNZ T2
16538  2D               DEC L
16539  20 EE            JRNZ T0
16541  C9               RET
16542  2A 84 40  SPCE:  LD HL,(SPEED)
16545  06 00     S0:    LD B,00
16547  10 FE     S1:    DJNZ S1
16549  2D               DEC L
16550  20 F9            JRNZ S0
16552  C9               RET
16553  3A 85 40  SPACE: LD A,(DELAY)
16556  C6 04            ADD A,04
16558  18 05            JR S2
16560  3A 85 40  ICG:   LD A,(DELAY)
16563  C6 02            ADD A,02
16565  CD 9E 40  S2:    CALL SPCE
16568  3D               DEC A
16569  20 FA            JRNZ S2
16571  C9               RET
16572  21 86 40  SEND:  LD HL, CWCH
16575  56        S3:    LD D,(HL)
16576  1E 04            LD E,04
16578  CB 02     S4:    RLC D
16580  CB 02            RLC D
16582  3E 03            LD A,03
16584  A2               AND D
16585  28 10            JR Z END
16587  21 DE 40          LD HL,SWIT
16590  85               ADD A,L
16591  6F               LD L,A
16592  0E 01            LD C,01
16594  E9               JP (HL)
16595  1D        BACK:  DEC E
16596  20 EC            JRNZ S4
16598  21 87 40         LD HL,CWCH+1
16601  18 E4            JR S3
16603  CD AD 40  END:   CALL ICG
16606  C9        SWIT:  RET
16607  0C               INC C
16608  0C               INC C
16609  CD 88 40  S5:    CALL TONE
16612  0D               DEC C
16613  20 FA            JRNZ S5
16615  CD 9E 40         CALL SPCE
16618  18 E7            JR BACK
16620  ED 4B 10 41 GETCH: LD, BC,(CHAR)
16624  79               LD A,C
16625  FE 0B            CP 0B
16627  38 13            JR C SP
16629  21 FC 40         LD HL,TABLE
16632  09               ADD HL,BC
16633  09               ADD HL,BC
16634  7E               LD A,(HL)
16635  32 86 40         LD (CWCH),A
16638  23               INC HL
16639  7E               LD A,(HL)
16640  32 87 40         LD (CWCH+1),A
16643  CD BC 40         CALL SEND
16646  18 03            JR EXIT
16648  CD A9 40  SP:    CALL SPACE
16651  ED 4B 10 41 EXIT: LD BC,(CHAR)
16655  C9               RET
16656  00 00     CHAR:  00 00
```

DEFINE TABLE = 16636

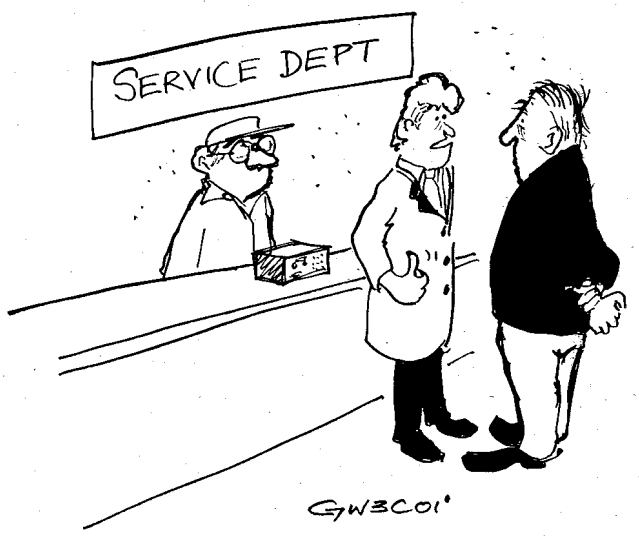
16658 — 16763 is the table of Morse characters. Each character takes two 8-bit bytes, *i.e.* 16558 and 16559 are the character " (quote). Two bits are used for each symbol — 11 for dot, 01 for dash and 00 for the end of the character.

References

[1] 'Machine code' is the instruction code of the Z80 chip, the microprocessor used in the ZX81. A useful book on the subject is "Programming the Z80" by Zaks, published by *Sybex*.

[2] The data port is described in an article "Control your own Substation" by D. E. Graham, in *Personal Computer World*, October 1981, p. 74.

Footnote: There are two versions of 8K ROM used in the ZX81. The old version contains an error which results in, for example, the result of PRINT SQR 0.25 being incorrect. The program as described will work on this ROM. The new ROM corrects this error, and with this version it will be necessary to POKE 16532, 70 before saving or running the program, otherwise it will go into an endless loop.



". . . he says the memory unit's got amnesia . . ."

September issue due to appear on Friday, August 27th.