TRS-80 MORSE CODE TRANSMIT & RECEIVE PROGRAM BY W4UCH VER 2.2

Detailed operating instructions are presented in Part 1 of the program. The main program is Part 2 that also includes a 5 page instruction summary that may be called anytime from the TRANSMIT MODE by operators new to the system. The program documentation that follows will not duplicate in detail the instructions included in Part 1, but is provided for those users who "wish-to-dig-deeper" into this BASIC program's logic, program flow, and layout.

It is written in Level II BASIC which allows transmit and receive code speeds up to the 25 word per minute area, yet most importantly allows the average user to follow the program logic and flow and make his/her modifications to the program very easy to accomplish which machine code would not allow.

The most unique aspect of this program is that it will both transmit and receive Morse code on any standard Level II TRS-80 Microcomputer (or most any microcomputer that uses 8K and up Microsoft Basic with modest changes) WITHOUT any peripheral or ancillary devices whatsoever. The cassette motor control relay K1 is used for the keying relay and the cassette EAR plug line for receiving Morse code audio of approximately 1 volt peak to peak derived from the station receiver's speaker terminals.

A unique software solution renders the TRS-80 flip-flop Z-24 "invisible" to incoming signals. Since the TRS-80 cassette control relay K1 will only handle VERY low power levels (about 6 volts at 400-500 mils) it is STRONGLY recommended that a 7406/7407 TTL buffer chip or Radio Shack #275-004 ($2.99) relay be used as a buffer between the TRS-80 and the station transmitter.

For gaining program operating experience, a low cost Radio Shack Morse Code Practice Oscillator (#20-005) may be used for both generating the Morse code for the TRS-80 to decode and print out (using the earphone jack on the code oscillator connected to the TRS-80's EAR plug), and by installing a subminiature Radio Shack phone jack on the code oscillator across the keying circuit, and having the TRS-80 subminiature cassette REMOTE plug plugged into it for TRS-80 Morse output.

PROGRAM ORGANIZATION:
The program consists of eight segments:
1. Initialization (CLEAR, DIMENSION & DEFINE INTEGERS)
2. Transmit Morse look-up table (1=dot and 2=dash)
3. Transmit Morse timing
4. "Q" signal - prepared messages (20+)
5. Code practice (alphabet, alphanumerics, all + punctucation)
6. Receive Morse decoding algorithm
7. 100 page automatic logbook/file for contests, etc.
8. Instruction summary (5 pages for newcomers)

INITIALIZATION SEGMENT:

This segment allows the user to choose either alphanumeric readout or Morse code readout on the video display, to choose transmit Morse code speed (receive speed is automatic), and reminds the operator that "left-arrow" calls up the instruction summary and the "CLEAR" key is the transmit/receive switch. It also defines as integers A to Z for optimum code speed, DIMENSIONS the Morse receive array, and CLEARS 2550 bytes for the automatic logbook/file. Lastly, it includes an error trapping function that may be deliberately invoked to obtain immediate return to the TRANSMIT mode.

TRANSMIT MORSE LOOK-UP TABLE:

Surprisingly, the fastest means of generating Morse code using the 12K Microsoft BASIC in the Level II TRS-80 (no machine code allowed to facilitate user comprehension) is the simple IF-THEN statement and look-up table which converts the alphanumeric/punctuation symbol to a number in which all 1's = dots and all 2's = dashes, for its equivalent Morse code character; i.e., A=12, B=2111, C=2121, etc. Each character is followed by a GOTO directive which directs the program to the transmit Morse timing segment to save time. Though Version 2.2's transmit Morse look-up table is given alphabetically and numerically for convenience, it may be further speeded up by rearranging the alphanumerics in the same order as the DATA table in the RECEIVE Morse decoding segment, if desired.

In the RECEIVE mode, the order approximates the most commonly used letters in the English language, in order of usage, which was the way Morse code has evolved to today's standard; i.e, "E" is the most commonly used letter, so it = a dit (1), and "Q" the most infrequently used letter so it = dah dah di dah (2212). If one were to transpose the 1's to 0's and the 2's to 1's you would have binary numbers = to the most frequently used alphabet characters in English in proper order. Give you any ideas?

In the TRANSMIT MODE all generally accepted Morse characters are provided plus EOM (end of message) and EOW (end of work) by using the "#" and "&" symbols, respectively. Should a keyboard character such as "$" or "%" or "@" which has no Morse equivalent be be entered via the keyboard inadvertently, an error-trapping subroutine skips it and the program awaits the next legal alphanumeric with a Morse equivalent. At the end of the look-up table the ASCII codes for up-arrow, left-arrow, down-arrow, right-arrow, and CLEAR key are scanned and the programm directed to the subroutines of: "Q" signal-message, auto-logbook, instruction summary, log book review, or RECEIVE mode as called by YOU, the operator and Chief-Pilot of this exquisitely wonderful machine.

TRANSMIT MORSE TIMING SEGMENT:

Here is the segment that accomplishes the job of translating the Morse character's 1's and 2's (it could just as well have been 0's and 1's) into properly timed dots and dashes with the correct timing intervals between each element and character via the LEN and MID string functions which allows the program to 'peel off' each element of a character, one at a time. The international standard of: dot = 1 time interval, dash = 3 times 1 dot interval, and space between dot/dash elements of a given character = 1 dot interval, is set by this segment. Tranmitting code speed is determined by multiplying each element that forms a Morse character by "S" which is the adjusted value for the desired code speed that the operator INPUT during program initialization.

Upon completion of each Morse character, this segment then directs the program to the operator selected input; i.e., back to the keyboard, "Q" signal/message subroutine, RECEIVE mode, logbook subroutines, or instruction summary. Prior to output on video, this segment determines whether alphanumerics or Morse code was selected. Morse printout via 1's and 2's is seldom used except for the first few training periods with those individuals just beginning to learn Morse.

The cassette motor control relay K1, on the bottom left side of the lower TRS-80 printed circuit board (tubular yellow plastic enclosure), is closed and opened as the keying relay via the OUT (port) 255 statement. If the user is skilled in printed circuit board work, it is a simple matter to install a normally closed mini-phone jack on the rear of the TRS-80 keyboard in series between relay K1 and the output of integrated circuit Z-41 for transmit keying. Thus allowing Z-41 to drive a separate relay such as the Radio Shack #275-004 for transmitter keying. This relay will handle 125 volts ac at 1 amp and is fast enough to follow the program up to about 25 words per minute. Conversely, a high speed 5 to 6 volts dc reed relay may be used which will follow this program up to about 40 words per minute. Above this speed, program execution time in BASIC becomes the limiting factor. By utilizing the excellent Mumford Micro Systems 3 speed TRS-80 clock modification, both TRANSMIT and RECEIVE modes may be increased an additional 50 percent.

"Q" SIGNAL - PREPARED MESSAGE SEGMENT:

Twenty prepared "Q" signal and message formats are given including: CQ, QTH, QRZ, QRX, QSL, QSY, QSY+, QSY-, QRM, QRN, QRS, QRQ, RST, QSL, 73, etc. There is no limit to the number of additional messages that may be added, except available memory. There is also a SPEED subcommand which allows the operator to change transmit code speed without reinitializing the program and losing the data stored in the automatic logbook funtion. This segment also allows the operator to select the type of TRANSMIT Morse code practice desired.

Code 1 = alphabet only, Code 2 = alphanumerics, Code 3 = alphanumerics + punctuation. Though the arrow syrbols are illustrated as reminders, they may only be used during the transmit or receive modes. This segment also uses the LEN and MID string functions of Level II BASIC for peeling off each letter, one at a time, for each prepared message. Each message is limited to a maximum of 240 bytes (string length), but by concatenating strings with appropiate software mods any message of any length may be transmitted. One final noteworthy subcommand included in this segment is the TEST subcommand. This function outputs the word PARIS with appropriate letter and word spacing standards so that the operator may time the number of words sent for 15 seconds, multiply by 4, and have his exact words per minute code speed calibration.

CODE PRACTICE SEGMENT:

This unique subroutine utilizes the random number generator incorporated in the Zilog Z-80 microprocessor to generate a number between 1 and 26 in the Code 1, alphabet only, code practice mode. By adding 59 to to the random number the ASCII character code for the alphabet from A to Z is generated and output, a letter at a time, in 5 letter code groups. Code 2, alphanumerics, is generated in much the same way by randomly generating a number from 1 to 47 and adding 48 to it to obtain the ASCII character code for both numbers and alpahbet. Since ASCII character codes 60, 61, 62, and 64 which equal less than, equal sign, greater than, and @, respectively, have no Morse code equivalents, they are trapped and not output.

Code 3, alphanumerics + punctuation is generated in much the same way. For brevity, the Morse double dash is displayed on video as a single dash, but for purists may be easily modified to a double dash if desired. Also, the normal 7 times dot length spacing after punctuation has been held to only 3 times dot length as it has been found in numerous Morse code training seesions that this convention speeds up the learning process. Spacing between each 5 letter group uses the international standard 7 times dot length for word spacing.

RECEIVE MORSE DECODING SEGMENT:

Utilizes an algorithm derived by the MIT Radio Club many years ago and improved upon by Robert Kurtz and the author. Its claim to fame is the method we developed to interface the TRS-80 with an ordinary communications receiever's speaker output that DOES NOT REQUIRE any ancillary/peripheral devices to work properly with Morse signals of S4 or stronger. Through sheer serendipity, cleverness, and lots of luck this subroutine makes the TRS-80 flip-flop Z-4 invisible to the approximately 1 volt peak to peak audio Morse signal coming from the station receiver's speaker terminals. This is done by re-setting flip-flop Z-4 every time the length of time is measured by the program to determine whether the Morse element is a dot, dash, or element space. With a good signal to noise ratio incoming signal (S4 or better), it will copy well sent

Morse up to 20 to 25 words per minute, which is about its upper limit due to BASIC (with standard clock) program execution time.

For operators working stations with "swing fists" (odd-ball dot/dash timing ratios), the RECEIVE MODE subroutine allows the operator to change these ratios by pressing "P" on the keyboard. This takes a bit of experimenting and experience, but after a few hours operating is quite easy to implement. For operating convenience, the FILE (auto-logbook) function may be called from both the transmit and receive modes.

There are many algorithms for decoing Morse code that may be written in Level II BASIC, but when the trade-offs between program length and execution time are evaluated, this version appears to be the best compromise. Z-4, an LM-3900 Norton operational amplifier in the TRS-80's cassette CASSIN input line is designed to serve as a pulse shaping and level adjusting network for the CLOAD function in normal TRS-80 operation. It works remarkably well for Morse code too, but may be improved a bit (for very weak signals) by adding an AVC/limiting amplifier between the TRS-80 and receiver speaker terminals as described by N6WA in the Sept. '79 issue of 73 magazine, pages 116-117, if an emitter follower 2N2222 transistor is added to match the TRS-80's 100 ohm input impedance.

FILE AND FILE-REVIEW SEGMENT:

This function is provided the operator to create a semiautomatic logbook with auto-sequencing for each entry. During initialization the program CLEARs 2550 bytes for this subroutine thus allowing only 25 bytes per entry if all 100 log entries were used. By all means CLEAR as many bytes as your installed memory will allow. This subroutine's most useful function is during CW (Morse code) amateur radio contests which is the reason for the auto-sequencing aspect of the program as time/speed are important. The FILE subroutine may called from both TRANSMIT and RECEIVE modes by pressing the 'right-arrow' on the keyboard. Each time it is called it will automatically advance to the next unused file where the call letters of the station worked, date, time, band, or what have you may be entered.

When the FILE REVIEW subroutine is called by pressing 'down-arrow' on the keyboard, the program will sequentially display 4 files per page (16 lines maximum if each of the 4 files is filled to capacity), each time the ENTER key is pressed. You do not have to review all 25 file pages (4 per page times 25 = 100 total) to return to the TRANSMIT mode, but may escape anytime by pressing 'break,' then '@,' and then ENTER. Here we deliberately induce an error and use the ON ERROR GOTO function to immediately put us back in the TRANSMIT MODE. This is a real time saver during CW contests.

At the end of a day's operation, or end of a contest, the file data may be saved on cassette or disk for permanent record and storage using the PRINT#-1, function described on page 3/10 (for cassette) of the Level II manual. If you plan to use this function frequently, by all means add the following lines to this program:
5000PRINT#-1,BA$:PRINT#-1,BB$:PRINT#-1,BC$:PRINT#-1,BD$ (etc). Remember that each print statement will only handle strings that TOTAL 240 bytes. This is why the PRINT#-1, is repeated for each string we wish to CSAVE. Add as many lines and strings as you wish to CSAVE and then press 'BREAK,' enter RUN 5000 (with the cassette turned "ON" and RECORD and PLAY depressed) whenever you wish to make a permanent record of your logbook entries.

INSTRUCTION SUMMARY SEGMENT:
One usually does not write instructions on how to use instructions. The program's instruction summary is provided basically for the user new to the system who does not wish to pickup a written instruction during system operation. It is called from the TRANSMIT MODE by pressing 'left-arrow.'

HINTS AND KINKS FOR SUCCESSFUL SYSTEM OPERATION:
Probably the most difficult challenge presented to the user of this Morse Code System (or ANY TRS-80 Morse Code Program) will be the problem of quieting down the RFI (radio frequency interference) generated by the TRS-80 itself.

Every little digital gate in the TRS-80 plus the nominal 10.6445 MHz crystal oscillator and all the clock dividers are each and every one a miniature spark coil transmitter, or at the very least act like one. Do not let this bamboozle or overwhelm you. We shall overcome if we follow a few not too difficult ground rules that will allow us to easily copy most any Morse signal that we can hear. After July 1, 1980 all new microcomputers will have to meet the FCC rules regarding spurious radiation levels. Till then, try these recipes to minimize the problem:
1. Use Radio Shack #15-1106 line filters on EACH component's power line after cutting each power cord to minimum length.
2. Physically separate the TRS-80 at least 6 feet from the station receiver.
3. Run good quality well shielded (not the cheapest you can buy) RG8/U SEPARATELY from the transmitter and receiver to your antenna.
4. Your station antenna should be AT LEAST 80 feet away from the TRS-80. Install T/R relay and broadband preamplifier AT the ANTENNA. This is the most important item of all.
5. If all else fails, turn-off expansion interface when operating and DISCONNECT cassette and interface cables at the keyboard. When the operating day is finished, CSAVE your auto-logbook BEFORE powering up the interface and printing out the logbook data. NOTE: shielding & grounding all TRS-80 cables helps too. DO NOT BLAME THIS PROGRAM IF IT WILL COPY MORSE FROM YOUR CODE PRACTICE OSCILLATOR, BUT NOT FROM YOUR RECEIVER.....I.E., BETTER SHIELDING IS NECESSARY. 'Gud luk.'
Note: See March '80 QST pages 17-20, "Computer Interference."

RICHCRAFT ENGINEERING LTD.
Phone: (716) 753-2654
DRAWER 1065
CHAUTAUQUA, N. Y. 14722

NOTE - NOTE - NOTE -
Also change LINE 59 to read:
DEFINT A-C,E-Z:------

W4UCH TRS-80 MORSE CODE PROGRAM MODIFICATION #1 - JULY 4, 1980

IF YOUR TRS-80 has the two I.C. E-Z Cassette Load Modification installed, the following changes to the RECEIVE section will GREATLY improve its accuracy up to about 20 WPM. Our thanks to Gene Steele-K5EVE in Orangefield, Texas for the suggestion. Please IGNORE line numbers as many programs are different, but IN THE SAME RELATIVE ORDER. Change lines with ARROW ------> .

```
168 REM "W4UCH/2 TRS-80 MORSE DECODER PROGRAM LINES 168 - 208
169 INPUT"CHARACTER TIMING: 5 NOMINAL";CC
170 INPUT"SPACE TIMING: 3 NOMINAL";SS:GOTO173
171 PRINT"RECEIVE MODE":RESTORE:CC=55:SS=3
172 FORN=1TO100:READA$(N):NEXTN
173 A=INP(255)
174 C$=INKEY$:IFC$="P"THENGOTO169
175 IFC$=CHR$(31)GOTO66:REM          TRANSMIT MODE LINE NUMBER
176 IFC$=CHR$(10)GOTO209:REM    AUTO FILE ROUTINE LINE NUMBER
----> 177 IFA<200THEN173
178 B=0
----> 179 IFA>200THENOUT255,0
180 A=INP(255):B=B+10
----> 181 IFA<200THENC=((5*C)+(2*B))/6:DO=2*DO:DA=2*DA:DO=DO+1:GOTO188
182 IFB<(.5*C)THEN179
183 DO=2*DO:DA=2*DA:DA=DA+1
----> 184 IFA>200THENOUT255,0
185 A=INP(255):B=B+10
----> 186 IFA>200THENGOTO184
187 C=((4*C)+B)/5
188 B=0
----> 189 IFA>200THENOUT255,0
190 A=INP(255):B=B+CC
----> 191 IFA>200THENGOTO178
192 IFB<(.5*C)THENGOTO189
193 GOSUB199
194 A=INP(255):B=B+SS
----> 195 IFA>200THENGOTO178
196 IFB<(2*C)THENGOTO194
197 PRINT" ";
198 GOTO173
199 DA=DA*2
200 D=DA+DO
201 IFD>100THEND=100
202 PRINTA$(D)
203 DA=0:DO=0
204 RETURN
205 DATA (no change)
206 DATA (no change)
207 DATA (no change)
208 DATA (no change)
```

NOTE: You will find Gene's changes a GREAT IMPROVEMENT. W4UCH