ISSUE NUMBER 14 $1.50 IN NORTH AMERICA ($2.50 elsewhere) OCTOBER 1976

BYTE

the small systems journal

PRINTED IN USA

A Morse Code Station Data Handler

Bruce Filgate
Digital Equipment Corporation
Components Group Engineering
One Iron Way
Marlborough MA 01752

For some time, there has been a need in amateur radio for a machine that could both decode and generate Morse code; in addition, the decoder had to be capable of automatically tracking varying received code speeds. Although Morse code keyboards have been around in the amateur radio field for some time, decoders have not been so readily available. Since hardwired logic can be difficult to modify, I decided to implement the coder and decoder in software. Since a low price was desirable and high performance was not required, I used Digital Equipment Corporation's MPS Starter Set. This is an Intel 8008-1 based product which DEC has been marketing to the commercial world. This article describes my implementation using MPS.

Implementation

The main program consists of a few subroutine calls to the main tasks, as shown in listing 1. As can be seen in the source code of the monitor, there are two principal routines that provide for Morse input and output; these two routines are INPEND and OTPUT respectively. The other routines within the main modules provide housekeeping and data manipulation for these IO drivers.

It should be noted, on consulting the full program listing, listing 2, that there are two data tables: an ASCII table and a Morse data table. The only restrictions on modifying these tables are that both tables be the same size and in the same sequence. For instance, the ASCII table could be changed to reflect a non-ASCII code so that the console could be other than an ASCII terminal.

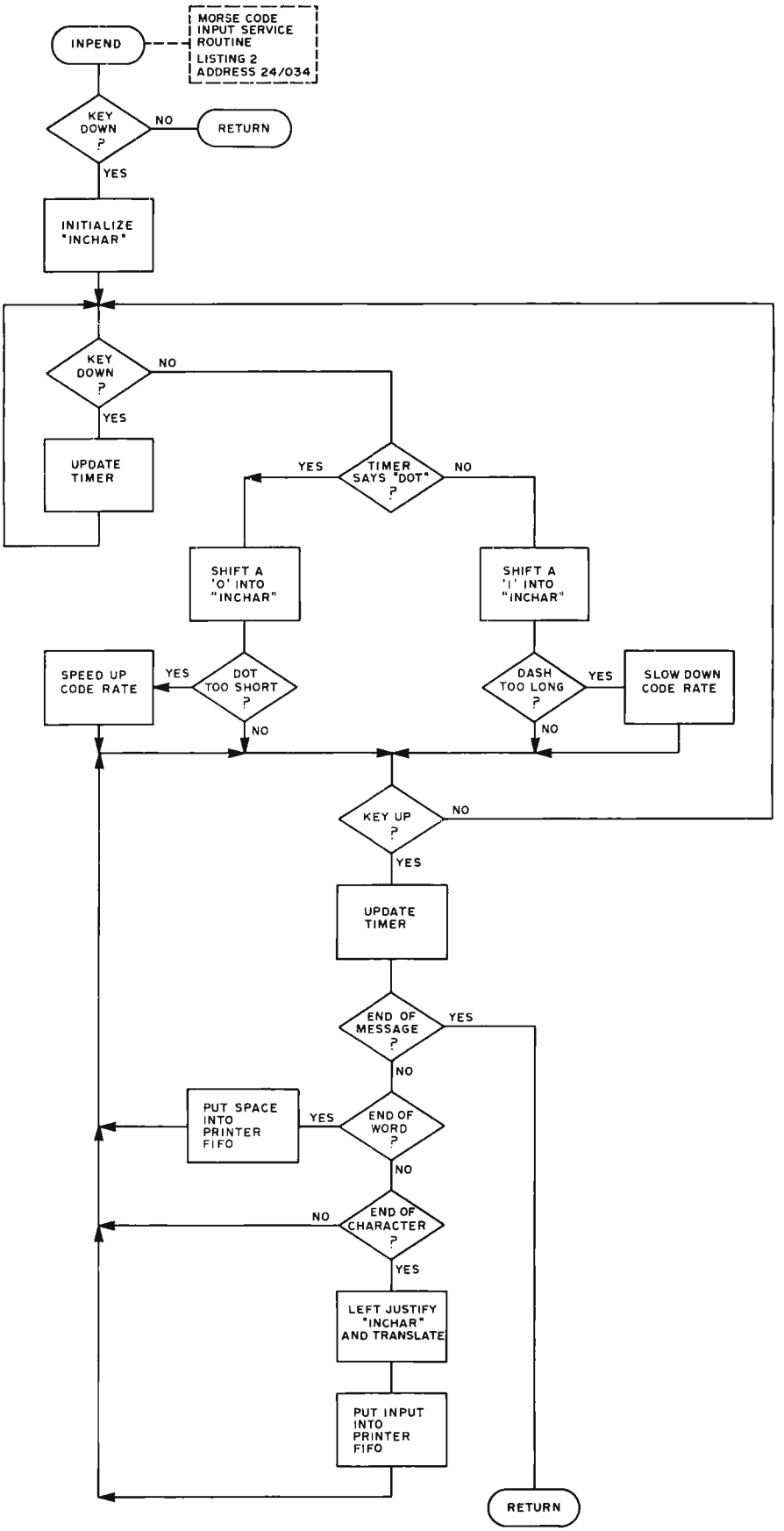
Listing 1: Monitor Entry and Supervisor Main Task. This listing shows the symbolic assembly language representation of the outer loop of the Morse code program. The detailed assembly is found in listing 2 along with the rest of the program.

```
RESTRT,   CAL    INPEND    try code input line;
          CAL    KYBD      try the keyboard task;
          CAL    PNTR      try the printer task;
          LHI    CMMND↑    test the mode byte;
          LLI    CMMND
          XRA              zero the A reg and flgs;
          ADM              mode byte to A reg and flgs;
          JFZ    CMMNDR    enter command mode;
          CAL    IDLE      non command character feedthrough;
          CAL    OUTPUT    anything morse to output?
          JMP    RESTRT    and loop (?) along;
```

Principle of Code Manipulation

A search of published literature available at the time this program was created showed an excellent method of representing Morse code in memory, particularly in eight bit wide memory. This method, found on page 13 of the July 1975 *QST* magazine, was adopted here for internal representation of Morse code. Binary ones represent dashes, binary zeros represent dots and a final binary one closes the character. For instance, the Morse character B(– • • •) would be represented by the octal number 210 (binary 10001000). Note that the binary strings are left justified with trailing zeros. It can be seen that if this representation is to be used to generate a Morse character B, the bit string should simply be shifted out to the left with each bit interpreted as a dash or a

Figure 1: Flow Chart Detail of INPEND, the Morse code input service routine. This routine, beginning at address 24/034 in listing 2, is responsible for tracking and adapting to the variations in speed of human generated Morse code inputs. This routine also detects end of character and end of word gaps between Morse inputs. The character outputs are translated and sent to the printer buffer maintained by the program.



dot until a final result in the shift register is 200 octal. Thus 200 octal remaining in the processor's accumulator is used to signify the end of a character.

To transmit an ASCII keyboard character out in Morse code, all that is required is to look up the ASCII character in the ASCII table and compute its relative location within the table. Once the relative location is calculated, it is necessary to look in the Morse table in the same relative location and pull the Morse equivalent of the ASCII character out of the table. Once this code is available, it is loaded into a register and shifted out to the left as dashes and dots until the register contains octal 200. Conversely, data being received in Morse code starts a character by preloading of a register with octal 200 and then the shifting in of a bit at a time of dash and dot information in ones and zeros. When an intercharacter delay time is finally recognized, the character is considered "closed" and the data that is in the register is left justified and looked up in the Morse code table. When it is found in the Morse code table, the same relative address within the ASCII table contains the ASCII character equivalent. (See figure 1 for a flow chart of the code input service routine.)

Morse Code Speed Determination

When code is being transmitted by the program, the code output speed is selected by a keyboard control sequence. This control sequence consists of the escape (ESC) key followed by a 'W' and then a second character obtained from table 1. Once the

Table 1: The list of ASCII character graphics and their equivalent speed settings when used with the speed change command. The command sequence is: escape character, W, speed character. Thus the sequence escape, W, + sets the speed to 18 words per minute.

Speed Character	Rate (wpm)
SP	120
!	89
"	63
#	48
$	44
%	37
'	27
)	22
+	18
/	14
3	12
>	7.6
?	7.2

Software Availability in Machine Readable Form

The software for this program in source and binary form has been submitted to the DECUS library. DECUS is the Digital Equipment Corporation User's Society, managed by a board of directors composed of DEC equipment users. Free membership in the society is open to users of DEC equipment. The group periodically publishes indices of currently available programs with abstracts.

The documentation for this program is listed under the number 8-801. Since it is expected that some users will want to modify this program, the source in ASCII is available on paper tape from the DECUS library. To order copies of the ASCII source tape or the binary tape, write to:

Program Libarian
DECUS
146 Main St
Maynard MA 01754

as of the time of writing, the following prices apply:

8-801	Binary paper tape	$2
8-801	ASCII source paper tape	$8
8-801	DECUS writeup	$1
8-801	Assembly listing	$10

output speed is selected by this method it will remain constant until a new value is selected.

Since the DEC M7341 processor card uses a variable speed clock, table 1 is calculated assuming that the clock be operated at the same speed as the clock on the author's system. If the clock for the processor on which the software is running is operated at a different rate, then table 1 must be recalculated.

If it is desired that the user's clock be set at the same speed as the clock found in the author's processor, a method of calibrating the clock is offered here: If the program is set up to transmit a sequence of dots, for instance the character "five" which is five dots sequentially, a string of "fives" will generate five dots followed by three dot times between characters, five more dots for the next "five" and so on. This, in consideration with the equation:

$$\text{Speed (wpm)} = \frac{\text{dots per minute}}{25} = \frac{}{2.4 * \text{dots per second}}$$

can be used to compute the effective code rate in words per minute.

The code input speed is never actually calculated but instead a rather heuristic tracking technique is used to update what

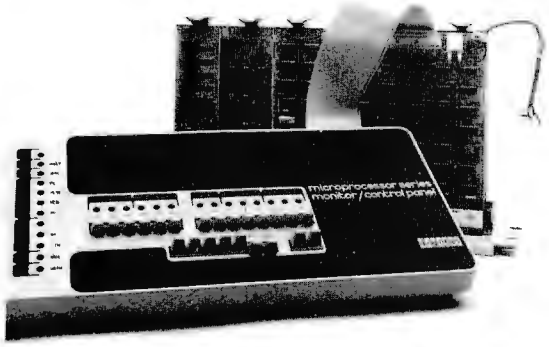


Photo 1: The Digital Equipment Corporation's MPS starter set used by the author for running the Morse code program. Further detail of the central processor board is shown in photo 2.

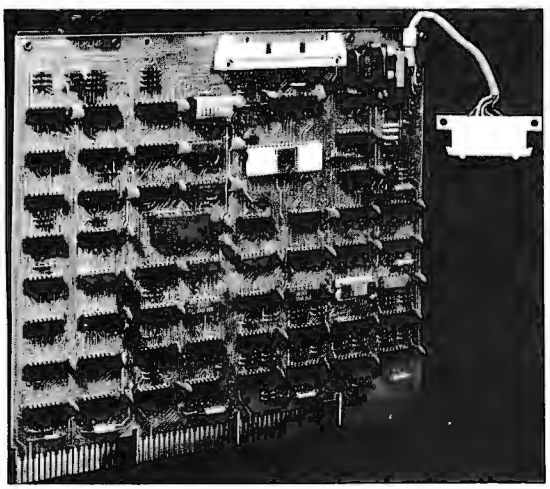


Photo 2: The Digital Equipment Corporation M7341 central processor board. The 8008 processor is socketed at the lower right. This board also includes a UART for serial interface and the random logic needed to buffer and drive an 8008.

Information about the Hardware Used

This program is designed to generate and decode Morse code. Although the program is intended to be executed on Digital Equipment Corporation's MPS M7341, it should also execute on almost any Intel 8008-1 based microprocessor with only slight modification. Photos 1 and 2 are supplied by Digital Equipment Corporation.

The following literature is available from:

Digital Equipment Corporation
Communications Services
444 Whitney St
Northboro MA 01532

General Interfacing Techniques for the M7341 Microprocessor Module
Interfacing The TU60 to the MPS M7341 Microprocessor
M7341 Processor Module Data Sheet
M7344-YA, −YB, −YC Read-Write Memory Module Data Sheet
M7345 Programmable Read Only Memory Data Sheet
M7346 External Event Detection Module Data Sheet
M7328 Evoke Decoder Module Data Sheet
M1501 Bus Input Interface Module Data Sheet
M1502 Bus Output Interface Module Data Sheet
Logic Handbook
KMP01A Microprocessor Series (MPS) Prewired Backplane Appl. Note
MR873 Microprocessor ROM Programmer Product Bulletin
1976 Direct Sales Catalog

the processor "judges" the current code rate to be. Therefore, the processor arbitrarily selects the initial code speed to be about 15 words per minute and, if it deems that the input code speed is other than that, a change will be made in the appropriate direction until the processor is able to synchronize against the incoming code speed. At this time, small changes in the code rate will be made to insure that the code speed remains within the tracking range. In addition to decoding the dot and dash times, the program must also be able to decode the times between characters, between symbols and even at the end of sentences. Additionally an arbitrary time is selected which is deemed to be an end of message; these times are set to be a function of a dot time. Thus, as the processor works to synchronize the code speed, all that is required is to keep track of a counter which represents the length of a dot. If the dot time is increased, thus decreasing the code speed, all the other times will be affected in a similar direction.

Keyboard Monitor

In general, characters typed on the keyboard are immediately translated to the Morse code bit string and then transmitted

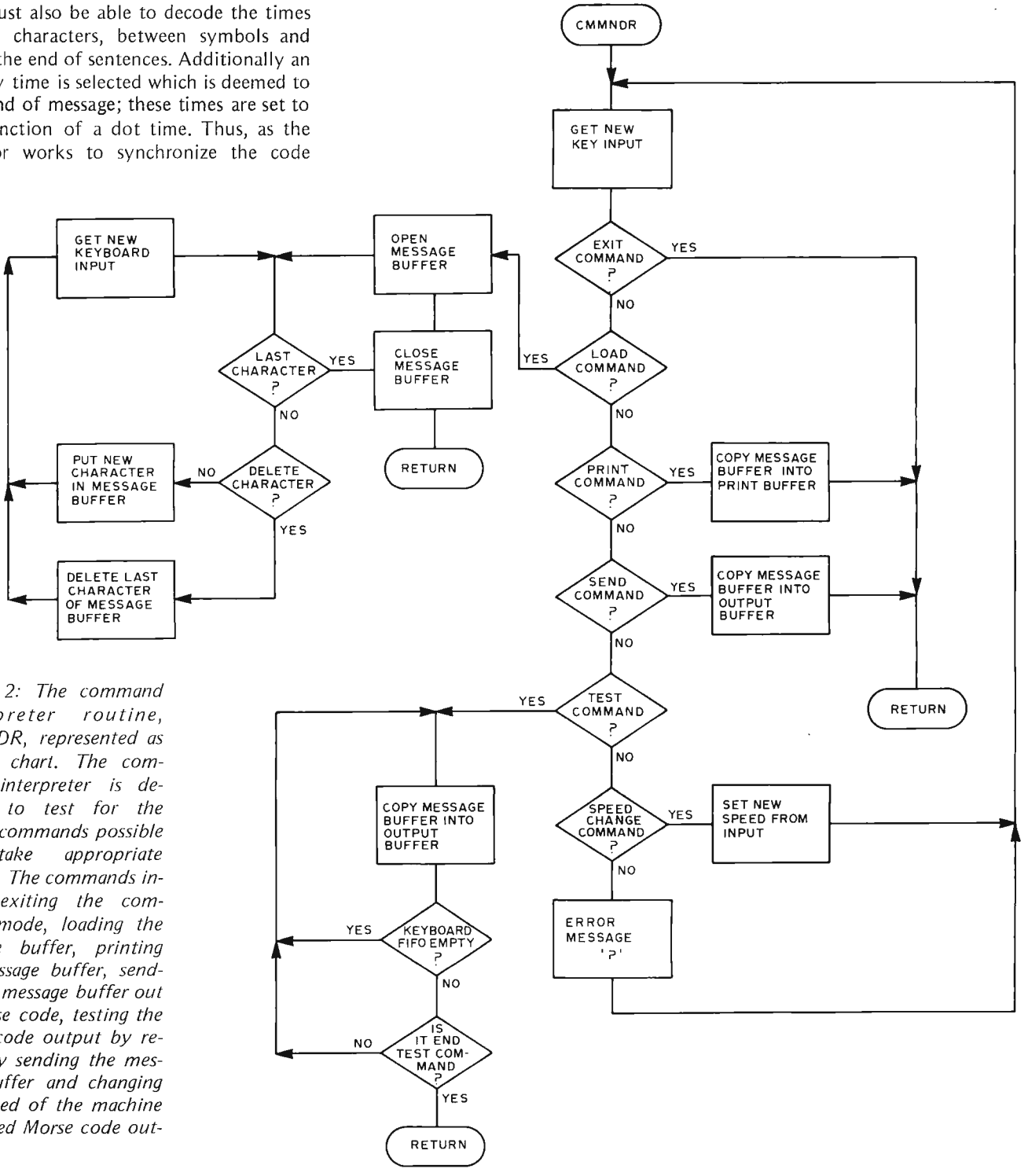


Figure 2: The command interpreter routine, CMMNDR, represented as a flow chart. The command interpreter is designed to test for the several commands possible and take appropriate actions. The commands include exiting the command mode, loading the message buffer, printing the message buffer, sending the message buffer out as Morse code, testing the Morse code output by repeatedly sending the message buffer and changing the speed of the machine generated Morse code outputs.

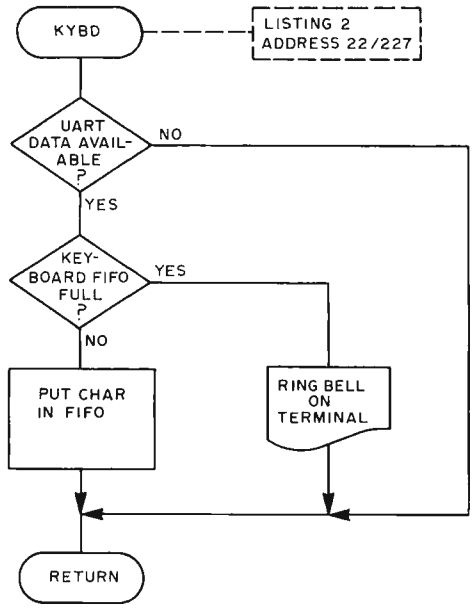


Figure 3: The keyboard service routine, KYBD, specified as a flow chart. This routine is simple: Look at the UART and see if data is available. If so, then stuff the data into the keyboard input buffer, or ring the bell and ignore if the buffer was full.

out in Morse code. There are some special commands interpreted as shown in figure 2 to modify this normal mode:

- <ESC> L is used to reload the message buffer until an escape (<ESC>). (LOAD command)
- <ESC> is used to return to normal keyboard mode. (EXIT command)
- <ESC> P causes the message buffer contents to be printed. (PRINT command)
- <ESC> S is used to ship the message buffer out, translated in Morse code. (SEND command)
- <ESC> T is a test to do <ESC> S until an escape is typed. (TEST command)
- <ESC> W loads a new rate constant for WPM output. A character obtained from table 1 sets the rate, and should follow the W. (Speed change command)
- The delete (rub out) can be used in a buffer loading mode to remove previous characters back to the beginning of the buffer.

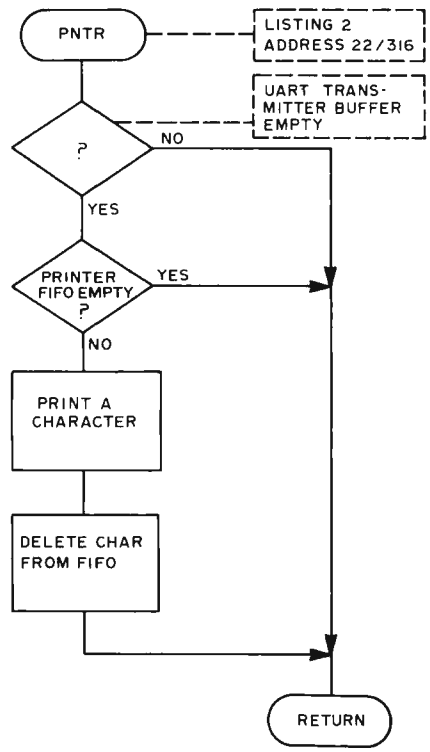


Figure 4: The printer service routine, PNTR, specified as a flow chart. This also is very straightforward: Look at the UART and see if it is ready to transmit. If so, then if the printer buffer is not empty send a character to the UART and remove the character from the printer buffer.

Software Buffer

This program uses overlaid IO to allow several operations to be done at the same time. First in, first out buffering is implemented to keep track of data transferred from or to the buffer locations. Four effective sources or destinations are involved: the message holding buffer, the code output buffer, the keyboard buffer (see figure 3 for the keyboard buffer service routine), and the printer buffer (see figure 4 for printer buffer service routine). Each buffer uses its first location as a byte count of the number of locations occupied by valid data. Since this is an eight bit machine, and single precision arithmetic is used, it is obvious that the maximum number of available data locations in a buffer can not exceed octal 377; any attempt by the user to exceed octal 377 locations or to exceed the maximum buffer size results in an error due to buffer overflow. When the program senses a buffer overflow condition, the terminal bell (or bell indicator) will be rung and the character that would have caused the overflow will be ignored. Since the user may choose to modify the sizes of these buffers, the following advice is offered: The message buffer should be as large as practically possible, followed by the keyboard buffer and then the printer buffer in priority.

Hardware Configuration

This piece of software has been designed and constructed so it can operate without modification on DEC Logic Products Starter Set 1 (KMP01 based). This starter set contains adequate memory, hardware, and interfacing to allow the software to execute properly. When executed on the starter set, the least significant bit of the input byte from input device 2 is used as the sense line for code input. The entire byte on output device 4 is used for code output (any particular bit on this output channel may be

Text continued on page 70

Listing 2: Complete Assembly of the Morse Code Program for an 8008. This listing was prepared using a cross assembler available to the author at Digital Equipment Corporation. The listing is reproduced here in its entirety, with an absolute origin picked for the hardware available to the author. The listing is well commented and includes a symbol table found at the end.

```
                        // THIS PROGRAM, CONSISTING OF A PROM SECTION OF
                        /SUBROUTINES & MAINLINE TASKS AND A RAM
                        /SECTION OF BUFFERS, GENERATES AND DECODES MORSE CODE.

                        /THE DECODING SECTION IS DESIGNED TO BE SELF TRACKING
                        /AS TO CHANGING SPEEDS.  DUE TO THIS SELF TRACKING
                        /FEATURE, A FEW ILLEGAL CODES MAY BE DECODED AS THE
                        /PROGRAM ATTEMPTS TO LOCK ONTO THE INPUT CODE SPEED.

                        /THIS PROGRAM IS DESIGNED TO RUN ON THE INTEL 8008-1 BASED
                        /MPS STARTER SET AVAILABLE FROM THE DEC COMPONENTS GROUP.



                        / WRITTEN 1-29-76 BY BRUCE FILGATE OF COMPONENTS
                        /APPLICATIONS ENGINEERING AT DIGITAL EQUIPMENT
                        /CORPORATION IN MARLBOUGH MASSACHUSETTS FOR THE
                        /LOGIC PRODUCTS MPS PRODUCT LINE



                        /WEIGHTING IS 1 DASH TO 3 DOTS.


                        /JUST A FEW WORDS ABOUT THE STACK STRUCTURE.....
                                /THE STACKS USE THE FIRST LOCATION AS A COUNT OF THE
                                /NUMBER OF OTHER USED LOCATIONS IN THE STACK.

                        /THE LSB OF THE INPUT BYTE FROM I/O CHANNEL #2 IS RESERVED FOR
                                /THE SENSE LINE FOR CODE INPUT.

                        /THE BYTE ON I/O CHANNEL #4 IS USED FOR CODE OUTPUT

                        /KEY SENSE FOR CODE INPUT AND OUTPUT IS GROUND FOR KEY DOWN
                                /CONDITION AND LOGIC HIGH FOR THE KEY UP CONDITION.


                        /*****************************************
                        /
                        / PROGRAM SHOULD BE STARTED AT STRT
                        /
                        /       PROGRAM MAY BE RESTARTED AT RESTRT
                        /
                        /*****************************************


                        /THE TERMINAL BELL WILL BE RUNG WHENEVER A BUFFER
                                /OVERFLOW IS CAUSED BY THE USER.  THE CHAR
                                /THAT WOULD HAVE CAUSED THE OVERFLOW IS TRAPPED
                                /AND DELETED.


                        /NORMALLY THE KEYBOARD DATA IS TRANSMITTED OUT TRANSLATED
                        /BUT ALT MODE (ESC) KEY USED FOR SPECIAL COMMANDS.
                                /L=RELOAD THE MESSAGE BUFFER UNTIL ESC
                                /ESC=RETURN TO NORMAL MODE
                                /P=PRINT THE MESSAGE BUFFER CONTENTS
                                /S=SHIP THE MESSAGE BUFFER TRANSLATED
                                /T=TEST BY DOING S UNTIL AN ESC IS TYPED
                                /W=LOAD BAUD CONSTANT FOR WPM OUTPUT
                                        /       SP      120 WPM
                                        /       !       89
                                        /       "       63
                                        /       #       48
                                        /       $       44
                                        /       %       37
                                        /       '       27
                                        /       )       22
                                        /       +       18
                                        /       /       14
                                        /       3       12
                                        /       >       7.6
                                        /       ?       7.2

                        /AND THE DELETE (RUBOUT) KEY IS USED TO EDIT BUFFER AS
                        /WELL AS REPRESENT THE ERROR CODE IN THE IMMEDIATE MODE.



                                /NEED A FEW MORE INSTRUCTIONS HERE...
                                OPDEF   SENSE#105#0     /READ THE SENSE LINE
                                OPDEF   READ#101#0      /SERIAL INPUT
                                OPDEF   PRINT#121#0     /SERIAL OUTPUT
                                OPDEF   STATUS#103#0    /SERIAL STATUS
                                OPDEF   OUTPUT#131#0    /ENCODED OUTPUT
                        /BUFFER SIZE SET UP
                                MSGSZ=377       /MESSAGE HOLDING
                                BUFOUT=60       /CODE OUTPUT
                                BUFSKY=377      /KEYBOARD
                                BUFSPN=60       /PRINTER

                        /TERMINAL DEPENDENT CONSTANTS
                                WIDTH=110       /PRINTER WIDTH IN OCTAL
                                CR=15           /CAR RET CHAR (ASCII CR=15
                                LF=12           /LINE FEED CHAR (ASCII LF=12)
                                W=127           /LOAD NEW SPEED CONSTANT (ASCII W=127)
                                ERCHAR=7        /CONSTANT FOR ERROR CHAR (ASCII BEL=7)
                                ESC=175         /ENTER COMMAND MODE (ASCII ESC=175)
                                L=114           /LOAD MESSAGE BUFFER (ASCII L=114)
                                P=120           /PRINT THE MESSAGE BUFFER (ASCII P=120)
                                S=123           /TRANSLATE&SEND MESSAGE (ASCII S=123)
                                T=124           /TEST DO S UNTIL ESC TYPED (ASCII T=124)
                                ESCSYM=44       /ECHO A $ FOR ESC
                                QUEST=77        /QUESTION MARK FOR BAD COMMAND
                                DELETE=177      /CHAR THAT REPRESENTS THE DELETE
                                DELSYM=134      /PRINTABLE CHAR FOR A DELETE
                                ETX=3           /CONTROL C EXIT TO STRT
                                C=103           /REQUIRED TO ECHO ETX, ^C
                                UPARRO=136      /REQUIRED TO ECHO ETX, ^C
                                BLANK=40        /ASCII SPACE CONSTANT


                                *20#120
                                /START UP TIME HOUSEKEEPING
20 120  056     STRT,   LHI     MSSGBF^ /CLEAR THE MESSAGE BUFFER
        026
20 122  066             LLI     MSSGBF
        044
20 124  370             LMA
20 125  056             LHI     BAUD^   /SET OUTPUT BAUD AT ABOUT 15 WPM
        024
20 127  066             LLI     BAUD
        301
20 131  076             LMI     35
        035
20 133  106             CAL     INCLH   /LIKEWISE WITH INPUT BAUD
        046
        021
20 136  076             LMI     35
        035
20 140  250     STRT1,  XRA             /RESTART FOR ^C AND CLEAR THE A REG
20 141  056             LHI     KYFIFO^ /CLEAR THE KYBD CHAR COUNT
        024
20 143  066             LLI     KYFIFO
        305
20 145  370             LMA
20 146  056             LHI     OTFIFO^ /CLEAR THE OUTPUT BUFFER
        025
20 150  066             LLI     OTFIFO
        364
20 152  370             LMA
20 153  056             LHI     CMMND^  /INITIALIZE THE MODE BYTE
        024
20 155  066             LLI     CMMND
        275
20 157  370             LMA
20 160  106             CAL     INCLH   /INIT THE CHAR COUNT
        046
        021
20 163  370             LMA             /SET FOR CRLF INITIALIZATION
20 164  056             LHI     PNFIFO^ /INITIALIZE THE PRINTER CHAR CNT
        025
20 166  066             LLI     PNFIFO
        304
20 170  370             LMA             /ZERO PRINTER CHAR COUNT
20 171  131             OUTPUT          /CLEAR THE TONE AND OTHER BITS


                        /MONITOR ENTRY AND SUPERVISOR MAIN TASK
20 172  106     RESTRT, CAL     INPEND  /TRY CODE INPUT LINE
        034
        024
20 175  106             CAL     KYBD    /TRY THE KEYBOARD TASK
        227
        022
20 200  106             CAL     PNTR    /TRY THE PRINTER TASK
        316
        022
20 203  056             LHI     CMMND^  /TEST THE MODE BYTE
        024
20 205  066             LLI     CMMND
        275
20 207  250             XRA             /ZERO THE A REG AND FLGS
20 210  207             ADM             /MODE BYTE TO A REG AND FLGS
20 211  110             JFZ     CMMNDR  /ENTER COMMAND MODE
        150
        021
20 214  106             CAL     IDLE    /NON COMMAND CHARACTER FEEDTHROUGH
        360
        023
20 217  106             CAL     OTPUT   /ANYTHING MORSE TO OUTPUT?
        230
        023
20 222  104             JMP     RESTRT  /AND LOOP (?) ALONG
        172
        020



                        /ASCII TABLE OF DATA
20 225  101     ASCTAB, BLOCK   32#101#1         /A THROUGH Z
        102
        103
        104
        105
        106
        107
        110
        111
        112
        113
        114
        115
        116
        117
        120
        121
        122
        123
        124
        125
        126
        127
        130
        131
        132
20 257  061             BLOCK   11#61#1          /1 THROUGH 9
        062
        063
        064
        065
        066
        067
        070
        071
20 270  060             DATA    60      /0
20 271  055             DATA    55      /-
20 272  056             DATA    56      /.
20 273  054             DATA    54      /,
20 274  077             DATA    77      /?
20 275  057             DATA    57      /SLASH
20 276  072             DATA    72      /:
20 277  050             DATA    50      /(
20 300  051             DATA    51      /)
20 301  047             DATA    47      /'
20 302  042             DATA    42      /"
20 303  012             DATA    12      /END OF MESSAGE(CR/LF)
20 304  012             DATA    12      /END OF WORK(CR/LF)
20 305  073     ASCEND, DATA    73      /;
```

```
                        /MORSE TABLE:1=DASH,0=DOT,WITH A 1 TO END THE CHAR
20 306  140  MORTAB, DATA  140    /A
20 307  210         DATA  210    /B
20 310  250         DATA  250    /C
20 311  220         DATA  220    /D
20 312  100         DATA  100    /E
20 313  050         DATA  50     /F
20 314  320         DATA  320    /G
20 315  010         DATA  10     /H

20 316  040         DATA  40     /I
20 317  170         DATA  170    /J
20 320  260         DATA  260    /K
20 321  110         DATA  110    /L
20 322  340         DATA  340    /M
20 323  240         DATA  240    /N
20 324  360         DATA  360    /O
20 325  150         DATA  150    /P

20 326  330         DATA  330    /Q
20 327  120         DATA  120    /R
20 330  020         DATA  20     /S
20 331  300         DATA  300    /T
20 332  060         DATA  60     /U
20 333  030         DATA  30     /V
20 334  160         DATA  160    /W
20 335  230         DATA  230    /X

20 336  270         DATA  270    /Y
20 337  310         DATA  310    /Z
20 340  174         DATA  174    /1
20 341  074         DATA  74     /2
20 342  034         DATA  34     /3
20 343  014         DATA  14     /4
20 344  004         DATA  4      /5
20 345  204         DATA  204    /6

20 346  304         DATA  304    /7
20 347  344         DATA  344    /8
20 350  364         DATA  364    /9
20 351  374         DATA  374    /0

20 352  206         DATA  206    /-
20 353  126         DATA  126    /.
20 354  316         DATA  316    /,
20 355  062         DATA  62     /?
20 356  224         DATA  224    /(SLASH)
20 357  342         DATA  342    /:
20 360  266         DATA  266    /(
20 361  266         DATA  266    /)
20 362  172         DATA  172    /'
20 363  112         DATA  112    /*
20 364  124         DATA  124    /END OF MESSAGE(CR/LF)
20 365  026         DATA  26     /END OF WORK(CR/LF)
20 366  252  MOREND, DATA  252    /;


                    /SUBROUTINE TO PUT DATA IN A GENERAL STACK
                    /STACK POINTER IN H&L, DATA IN B, BUFFER SIZE IN C
                    /RETURNS WITH A=0 IF NO ERROR, =ERCHAR IF ERROR
20 367  307  ENTPAK, LAM             /CHAR COUNT TO A
20 370  021         DCC             /COMPUTE CHARACTER LOCATIONS IN BUFFER
20 371  272         CPC             /DON'T OVERFLOW THE BUFFER
20 372  150         JTZ     ERROFL  /FULL...
             011
             021
20 375  004         ADI     1       /BUMP THE COUNT
             001
20 377  370         LMA             /CHAR COUNT UPDATE TO MEM
21 000  206         ADL             /LOW POINTER ADDED TO A
21 001  360         LLA             /GET L VALUE UPDATED TO L REG
21 002  100         JFC     OK      /IF A CARRY, FIX THE H
             006
             021
21 005  050         INH             /FIX H REG
21 006  371  OK,    LMB             /CHAR TO MEMORY
21 007  250         XRA             /CLEAR THE A REG
21 010  007         RET             /DONE
21 011  004  ERROFL, ADI     1       /SET A REG NON ZERO ERR RETURN
             001
21 013  007         RET             /SYSTEM ERROR, SYSTEM MUST FIX!!!!


                    /POP SUBROUTINE: ENTERED WITH POINTER IN H&L, SIZE IN B
21 014  307  POP,   LAM             /GET COUNTER TO A REG
21 015  024         SUI     1       /DECREMENT
             001
21 017  370         LMA             /RESTORE THE NEW COUNTER
21 020  104         JMP     POPY    /GET INTO POPLOP LOOP
             036
             021

                    /LOOP FOR POP
21 023  106  POPLOP, CAL     INCLH   /POINT AT CHAR TO POP
             046
             021
21 026  327         LCM             /GET CHAR TO C REG
21 027  106         CAL     DCRLH   /POINT TO NEW LOCATION
             052
             021
21 032  372         LMC             /PUT CHAR IN MEMORY
21 033  301         LAB             /B TO A REG
21 034  024         SUI     1       /SUBTRACT 1
             001
21 036  053  POPY,  RTZ             /DONE?
21 037  310         LBA             /RESTORE B REG
21 040  106         CAL     INCLH   /RECOVER FROM THE DECREMENT POSITION
             046
             021
21 043  104         JMP     POPLOP  /NEXT PAIR POP
             023
             021


                    /SUBROUTINE TO INCREMENT THE H AND L REGS
21 046  060  INCLH, INL             /BUMP THE L
21 047  013         RFZ             /RETURN IF NO CARRY
21 050  050         INH             /BUMP THE H ON A CARRY
21 051  007         RET             /ALL DONE


                    /SUBROUTINE TO DECREMENT THE H AND L REGS
21 052  306  DCRLH, LAL             /L TO A
21 053  024         SUI     1       /DECREMENT THE A
             001
21 055  360         LLA             /RETURN IF NO BORROW
21 056  003         RFC             /NO BORROW, THEN RETURN
21 057  051         DCH             /FIX THE H AFTER A BORROW
21 060  007         RET             /DONE


                    /SUBROUTINE TO WAIT A UNIT CODE TIME, DESTROYS A,B,C
21 061  056  TICK,  LHI     BAUD^   /POINT AT CONST
             024
21 063  066         LLI     BAUD
             301
21 065  026         LCI     50      /MULTIPLIER CONSTANT
             050
21 067  317  WAIT2, LBM             /CONST TO B
21 070  011  WAIT1, DCB             /COUNT IT DOWN DELAY
21 071  110         JFZ     WAIT1
             070
             021
21 074  021         DCC             /MULTIPLY IT
21 075  110         JFZ     WAIT2
             067
             021
21 100  106         CAL     KYBD    /OVERLAP WITH KEYBOARD INPUT
             227
             022
21 103  106         CAL     PNTR    /LIKEWISE WITH THE PRINTER
             316
             022
21 106  007         RET             /DELAY OVER!!!!


                    /DELAY, USED FOR 8 SLICE DECODING OF INPUT CODE
21 107  056  TICK1, LHI     BAUD1^  /POINT AT CONSTANT
             024
21 111  066         LLI     BAUD1
             302
21 113  026         LCI     5       /8 TIMES FASTER THAN TICK
             005
21 115  104         JMP     WAIT2   /FINISH TICK1 IN TICK ROUTINE
             067
             021


                    /SUBROUTINE TO GENERATE A DOT AND POST SPACE, DESTROYS A,B,C,H,L
21 120  006  DOT,   LAI     377     /SET ALL BITS IN THE A REG
             377
21 122  131         OUTPUT          /TURN ON THE KEY (DOWN)
21 123  104         JMP     FINDOT  /FINISH THE DOT IN THE DASH ROUTINE
             137
             021


                    /SUBROUTINE GENERATES DASH ITS POST SPACE, DESTROYS A,B,C,H,L
21 126  006  DASH,  LAI     377     /SET ALL BITS IN THE A REG
             377
21 130  131         OUTPUT          /KEY DOWN
21 131  106         CAL     TICK    /DASH
             061
             021
21 134  106         CAL     TICK
             061
             021
21 137  106  FINDOT, CAL     TICK    /ENTERED HERE TO FINISH A DOT
             061
             021
21 142  250         XRA             /CLEAR THE A REG
21 143  131         OUTPUT          /KEY UP
21 144  106         CAL     TICK
             061
             021
21 147  007         RET


                    /MONITOR TASK SUBROUTINE FOR HANDLING COMMANDS FROM THE KEYBOARD
21 150  106  CMMNDR, CAL     UNPAK   /GET CHAR FROM KEYBOARD
             164
             022
21 153  150         JTZ     CMMNDR  /IF NO CHAR, WAIT...
             150
             021
21 156  074         CPI     ESC     /WAS IT ANOTHER ESC?
             175
21 160  150         JTZ     CLRMD   /EXIT ON ESC
             153
             022
21 163  074         CPI     L       /WAS IT A LOAD?
             114
21 165  150         JTZ     LDNXT   /YES, GO LOAD A MESSAGE
             224
             021
21 170  074         CPI     P       /WAS IT A PRINT?
             120
21 172  150         JTZ     PRT     /YES GO PRINT THE MESSAGE BUFFER
             317
             021
21 175  074         CPI     S       /WAS IT A SEND THE BUFFER?
             123
21 177  150         JTZ     SNDNX   /YES, SHIP OUT THE MESSAGE
             060
             022
21 202  074         CPI     T       /WAS IT A TEST?
             124
21 204  150         JTZ     TEST    /YES, SEND THE BUFFER UNTIL ESC IS TYPED
             074
             022
21 207  074         CPI     W       /WAS IT A NEW WPM CONSTANT?
             127
21 211  150         JTZ     WPM     /YES, LOAD NEXT CHAR AS THE CONST
             133
             022

                    /IF HERE A BAD COMMAND
21 214  016         LBI     QUEST   /QUESTION MARK
             077
21 216  106         CAL     PPAK    /TO THE PRINTER FIFO
             304
             022
21 221  104         JMP     CMMNDR  /TRY FOR A VALID COMMAND CHAR
             150
             021

                    /ROUTINE TO LOAD THE MESSAGE BUFFER
21 224  056  LDNXT, LHI     MSSGBF^ /POINT AT THE CURRENT CHAR POINTER
             026
21 226  066         LLI     MSSGBF
             044
```

```
21 230  076          LMI     0       /ZERO THE COUNT
        000
21 232  106  LDNXT1, CAL     UNPAK   /GET KEYBOARD CHAR...
        164
        022
21 235  150          JTZ     LDNXT1  /WAIT FOR DATA
        232
        021
21 240  074          CPI     ESC     /END OF THE INPUT MESSAGE?
        175
21 242  150          JTZ     CLRMD   /EXIT END OF MESSAGE
        153
        022
21 245  310          LBA             /CHAR TO B REG FOR ENTPAK
21 246  056          LHI     MSSGBF^ /POINT AT THE MESSAGE BUFFER
        026
21 250  066          LLI     MSSGBF
        044
21 252  074          CPI     DELETE  /DELETE COMMAND?
        177
21 254  110          JFZ     LDNXT2  /NO
        272
        021
21 257  250          XRA             /YES, CLEAR THE A REG AND FLGS
21 260  207          ADM             /GET COUNT TO A AND FLGS
21 261  150          JTZ     LDNXT1  /BUFFER EMPTY, A NO NO
        232
        021
21 264  034          SBI     1       /DECREMENT COUNTER
        001
21 266  370          LMA             /RETURN COUNT TO MEM
21 267  104          JMP     LDNXT1  /LOOP
        232
        021
21 272  026  LDNXT2, LCI     MSGSZ   /BUFFER SIZE SET UP
        377
21 274  106          CAL     ENTPAK  /CHAR TO BUFFER
        367
        020
                /TEST FOR BUFFER OVERFLOW
21 277  112          CFZ     WHOOP   /TELL USER BUFFER FULL
        305
        021
21 302  104          JMP     LDNXT1  /LOOP UNTIL ESC
        232
        021


        /SUBROUTINE FOR USER ERROR INDICATION
21 305  103  WHOOP,  STATUS          /YES, GET PRINTER STATUS
21 306  044          NDI     20      /TBMT MASK
        020
21 310  150          JTZ     WHOOP   /WAIT
        305
        021
21 313  006          LAI     ERCHAR  /SET ERROR INDICATION
        007
21 315  121          PRINT
21 316  007          RET


        /ROUTINE TO PRINT THE MESSAGE BUFFER
21 317  056  PRT,    LHI     SOH^    /SET UP PRINT SUBMODE
        024
21 321  066          LLI     SOH
        300
21 323  076          LMI     0
        000
21 325  106          CAL     DMPSUB  /PRINT THE MESSAGE BUFFER
        333
        021
21 330  104          JMP     CLRMD   /EXIT TO THE SUPERVISOR
        153
        022


        /SUBROUTINE TO MOVE THE MESSAGE BUFFER CONTENTS TO LOCATION
        /DEFINED BY THE SOH LOCATION.  0=PRINTER    1=SENDER
21 333  056  DMPSUB, LHI     MSSCNT^ /POINT AT TEMP CHAR POINTER
        024
21 335  066          LLI     MSSCNT
        277
21 337  076          LMI     0       /CLEAR THE POINTER
        000
21 341  106  PRT1,   CAL     PNTR    /TRY TO FINISH THE PRINTING
        316
        022
21 344  106          CAL     KYBD    /TRY TO FINISH THE KEYBOARD INPUT
        227
        022
21 347  106          CAL     OTPUT   /TRY TO FINISH THE TRANSMISSION
        230
        023
21 352  056          LHI     MSSGBF^ /IS THERE A MESSAGE?
        026
21 354  066          LLI     MSSGBF  /FIND CHAR CNTR AND CHECK FOR NON ZERO
        044
21 356  250          XRA             /CLEAR THE A REG
21 357  207          ADM             /ADD IN THE CHAR COUNT
21 360  053          RTZ             /NO MESSAGE, EXIT
21 361  056          LHI     MSSCNT^ /CHECK COUNT ON XFER CHARS
        024
21 363  066          LLI     MSSCNT
        277
21 365  277          CPM             /BUFFER ALL XFERED?
21 366  053          RTZ             /EVERYTHING XFERED, EXIT
21 367  317          LBM             /STILL HERE, BUMP THE CHAR COUNT
21 370  010          INB
21 371  371          LMB
                /FETCH THE CHAR FROM THE MESSAGE BUFFER
21 372  006  PR4,    LAI     MSSGBF  /COMPUTE POINTERS
        044
21 374  207          ADM             /ADD IN THE BUFFER OFFSET
21 375  360          LLA             /SET UP THE L, H YET TO GO
21 376  056          LHI     MSSGBF^ /H SET IF NO CARRY FROM THE L
        026
22 000  100          JFC     PR2     /NO CARRY
        004
        022
22 003  050          INH             /FIX FOR THE L CARRY
22 004  317  PR2,    LBM             /CHAR TO THE B REG
22 005  250          XRA             /CLEAR THE A REG
22 006  056          LHI     SOH^    /GET THE SUBMODE
        024
22 010  066          LLI     SOH
        300
22 012  207          ADM             /SUBMODE IN THE A REG AND FLGS
22 013  110          JFZ     SOH1    /SEND SUBMODE
        024
        022
```

```
22 016  106          CAL     PPAK    /PRINT SUBMODE
        304
        022
22 021  104          JMP     PR3
        035
        022
22 024  056  SOH1,   LHI     OTFIFO^ /POINT AT OUTPUT BUFFER
        025
22 026  066          LLI     OTFIFO
        364
22 030  026          LCI     BUFOUT  /BUFFER SIZE TO C REG
        060
22 032  106          CAL     ENTPAK  /XFER TO THE BUFFER
        367
        020
22 035  150  PR3,    JTZ     PRT1    /LOOP IF NO ERROR
        341
        021
22 040  106          CAL     PNTR    /YES, OVERLAY THE I/O
        316
        022
22 043  106          CAL     KYBD
        227
        022
22 046  106          CAL     OTPUT
        230
        023
22 051  056          LHI     MSSCNT^ /RESET THE H REG FOR ERROR RECOVERY
        024
22 053  066          LLI     MSSCNT
        277
22 055  104          JMP     PR4     /TRY THE CHAR AGAIN
        372
        021


        /ROUTINE TO SHIP OUT THE BUFFER IN CODE
22 060  056  SNDNX,  LHI     SOH^    /SET UP SEND MODE FOR
        024
22 062  066          LLI     SOH     /DMPSUB ROUTINE
        300
22 064  076          LMI     1
        001
22 066  106          CAL     DMPSUB  /SEND THE MESSAGE BUFFER
        333
        021
22 071  104          JMP     CLRMD   /EXIT
        153
        022


        /ROUTINE TO TEST OUTPUT, SHIP UNTIL ESC IS TYPED
22 074  056  TEST,   LHI     SOH^    /SET UP FOR SEND MODE
        024
22 076  066          LLI     SOH
        300
22 100  076          LMI     1
        001
22 102  106          CAL     DMPSUB  /SEND THE BUFFER
        333
        021
22 105  106          CAL     UNPAK   /CHAR FROM KEYBOARD
        164
        022
22 110  150          JTZ     TEST    /NOTHING YET, DO IT AGAIN
        074
        022
22 113  074          CPI     ESC     /ESC?
        175
22 115  150          JTZ     CLRMD   /YES, EXIT
        153
        022
22 120  016  TEST1,  LBI     QUEST   /ILLEGAL CHAR FOR THIS
        077
22 122  106          CAL     PPAK    /NOTIFY THE USER
        304
        022
22 125  110          JFZ     TEST1   /IF OVERFLOW, TRY AGAIN
        120
        022
22 130  104          JMP     TEST    /AND LOOP....
        074
        022


        /CODE TO LOAD A NEW WPM CONSTANT INTO BAUD
22 133  106  WPM,    CAL     UNPAK   /GET CHAR FROM KEYBOARD
        164
        022
22 136  150          JTZ     WPM     /WAIT FOR CHAR
        133
        022
22 141  044          NDI     37      /MASK FOR 5 VALID BITS
        037
22 143  002          RLC             /MULTIPLY BY 2
22 144  064          ORI     1       /SET THE LSB
        001
22 146  056          LHI     BAUD^   /POINT AT BAUD LOCATION
        024
22 150  066          LLI     BAUD
        301
22 152  370          LMA             /CONSTANT TO BAUD LOCATION
        /FALL THROUGH TO CLRMD

        /CLEAR THE FLAGS AND EXIT TO SUPERVISOR
22 153  056  CLRMD,  LHI     CMMND^  /ZERO THE MODE BYTE
        024
22 155  066          LLI     CMMND
        275
22 157  076          LMI     0
        000
22 161  104          JMP     RESTRT  /TO SUPERVISOR
        172
        020


        /ROUTINE TO GET CHAR FROM KEYBOARD FIFO
                /RETURNS WITH CHAR IN A REG, A REG=0 IF NO CHAR
22 164  106  UNPAK,  CAL     PNTR    /TRY TO FINISH PENDING PRINTING
        316
        022
```

```
22 167  106          CAL     KYBD    /KEYBOARD HAPPY?
        227
        022
22 172  106          CAL     OTPUT   /OVERLAY THE CODE OUTPUT
        230
        023
22 175  056          LHI     KYFIFO^ /POINT AT KEYBOARD STACK
        024
22 177  066          LLI     KYFIFO
        305
22 201  250          XRA             /CLEAR THE A REG
22 202  207          ADM             /CHAR COUNT TO A REG
22 203  053          RTZ             /IF EMPTY, RETURN A REG=0
22 204  106          CAL     INCLH   /POINT AT CHAR
        046
        021
22 207  347          LEM             /GET CHAR TO E REG TEMP
22 210  106          CAL     DCRLH   /POINT AT KEYBOARD FIFO
        052
        021
22 213  106          CAL     POP     /OUT OF FIFO, RETURNS A=0
        014
        021
22 216  204          ADE             /CHAR TO A REG AND FLGS
22 217  007          RET             /RETURN WITH A REG=CHAR

                /SUBROUTINE TO WAIT FOR THE TBMT FLAG
22 220  103  WAITMT, STATUS          /GET TBMT FLAG
22 221  044          NDI     20
        020
22 223  150          JTZ     WAITMT  /WAIT....
        220
        022
22 226  007          RET             /OK, HAVE TBMT

                /KEYBOARD HANDLER SUBROUTINE
22 227  103  KYBD,   STATUS          /GET THE SERIAL LINE STATUS
22 230  044          NDI     40      /MASK
        040
22 232  053          RTZ             /NEXT TASK
                /PUT KEYBOARD CHARACTER IN KYBD FIFO
22 233  101          READ            /GET CHAR FORM KEYBOARD TO A REG
22 234  044          NDI     177     /GET RID OF ASCII PARITY BIT
        177
22 236  074          CPI     ETX     /CONTROL C
        003
22 240  110          JFZ     NETX    /NO
        267
        022
22 243  250          XRA             /YES, CLEAR OUTPUT
22 244  131          OUTPUT
22 245  106          CAL     WAITMT  /SEND OUT ^C
        220
        022
22 250  006          LAI     UPARRO
        136
22 252  121          PRINT
22 253  106          CAL     WAITMT  /WAIT FOR TBMT
        220
        022
22 256  006          LAI     C
        103
22 260  121          PRINT
22 261  106          CAL     WAITMT  /WAIT FOR TBMT
        220
        022
22 264  104          JMP     STRT1   /GO RESTART FROM ALMOST ZERO
        140
        020
22 267  056  NETX,   LHI     KYFIFO^ /POINT AT KYFIFO
        024
22 271  066          LLI     KYFIFO
        305
22 273  310          LBA             /CHAR TO B REG
22 274  026          LCI     BUFSKY  /BUFFER SIZE TO C REG
        377
22 276  106          CAL     ENTPAK  /PUT CHAR IN BUFFER
        367
        020
22 301  112          CFZ     WHOOP   /IF OVERFLOW, TELL THE USER
        305
        021
                /FALL THROUGH AND RETURN IN NEXT ROUTINE

                /PUT CHAR IN PRINTER FIFO FROM B REG
                /RETURNS WITH A REG =0 IF NO ERROR, =ERCHAR IF ERROR
22 304  056  PPAK,   LHI     PNFIFO^ /POINT AT PNFIFO
        025
22 306  066          LLI     PNFIFO
        304
22 310  026          LCI     BUFSPN  /BUFFER SIZE TO C REG
        060
22 312  106          CAL     ENTPAK  /PUT CHAR IN FIFO
        367
        020
22 315  007          RET
                /END OF THE KEYBOARD HANDLER TASK

                /PRINTER HANDLER SUBROUTINE TASK
22 316  103  PNTR,   STATUS          /GET THE PRINTER STATUS
22 317  044          NDI     20      /MASK FOR TBMT
        020
22 321  053          RTZ             /IF BUSY, TRY SOMETHING ELSE
22 322  056          LHI     TWIDTH^ /FIND PRINT POSITION
        024
22 324  066          LLI     TWIDTH
        276
22 326  250          XRA             /CLEAR THE A REG
22 327  207          ADM             /COUNT TO FLGS AND A REG
22 330  110          JFZ     PRT2    /NO LINE OVERFLOW
        347
        022
22 333  006          LAI     CR      /LINE OVERFLOW, FIX IT
        015
22 335  121          PRINT
22 336  106          CAL     WAITMT  /WAIT FOR TBMT
        220
        022
22 341  006          LAI     LF
        012
22 343  121          PRINT
22 344  076          LMI     WIDTH   /RESET PRINT POSITION COUNT
        110
22 346  007          RET
22 347  056  PRT2,   LHI     PNFIFO^ /POINT AT CHAR COUNT
        025
22 351  066          LLI     PNFIFO
        304
22 353  250          XRA             /CLEAR THE A REG
22 354  207          ADM             /CHAR COUNT TO A REG AND FLGS
22 355  053          RTZ             /NOTHING TO PRINT, NEXT TASK
        /IF HERE THERE IS PRINTING TO BE DONE!!!!!!!
                /POINT AT CHAR TO PRINT AND PRINT IT
22 356  106          CAL     INCLH
        046
        021
22 361  347  NXTPNT, LEM             /CHAR TO E REG TEMP
                /NOW UPDATE THE CHAR COUNT
22 362  066          LLI     PNFIFO  /POINT AT CHAR COUNT
        304
22 364  056          LHI     PNFIFO^
        025
22 366  106          CAL     POP     /RIPPLE THE FIFO
        014
        021
22 371  056          LHI     TWIDTH^ /UPDATE PRINT POSITION
        024
22 373  066          LLI     TWIDTH
        276
22 375  317          LBM             /COUNT TO B
22 376  011          DCB             /-1
22 377  371          LMB             /BACK TO MEMORY
23 000  304          LAE             /CHAR BACK TO A REG
23 001  074          CPI     LF      /IS IT A LF?
        012
23 003  150          JTZ     INCRLF  /YES, INSERT A CRLF
        031
        023
23 006  074          CPI     CR      /IS IT A CR?
        015
23 010  150          JTZ     INCRLF  /YES, INSERT A CRLF
        031
        023
23 013  074          CPI     DELETE  /IS IT A DELETE?
        177
23 015  150          JTZ     DEL     /YES, INSERT A BACKSLASH
        034
        023
23 020  074          CPI     ESC     /IS IT AN ESC?
        175
23 022  110          JFZ     PNT1    /NO
        027
        023
23 025  006          LAI     ESCSYM  /YES, SUBSTITUTE A PRINTABLE CHAR
        044
23 027  121  PNT1,   PRINT           /PRINT THE CHAR
23 030  007          RET             /DONE, PRINTED A CHAR
23 031  076  INCRLF, LMI     0       /SET FOR CRLF NEXT
        000
23 033  007          RET
23 034  006  DEL,    LAI     DELSYM  /SUBSTITUTE A PRINTABLE CHARACTER
        134
23 036  121          PRINT
23 037  007          RET             /END TO THE PRINTER TASK

                /THIS SUBROUTINE TRANSLATES A REG TO OUTPUT MODE
                /COMPUTE THE DISPLACEMENT IN ASCTAB
23 040  056  XLATER, LHI     ASCTAB^ /POINT AT ASCTAB
        020
23 042  066          LLI     ASCTAB
        225
23 044  277  THISIT, CPM             /IS THIS THE CHAR?
23 045  150          JTZ     CONVT   /GO CONVERT THE CHAR
        074
        023
23 050  106          CAL     INCLH   /TRY NEXT CHAR
        046
        021
23 053  310          LBA             /SAVE THE CHAR IN B REG TEMPORARILY
23 054  006          LAI     ASCEND^ /GET HIGH LIMIT TO A FOR COMPARE
        020
23 056  275          CPH             /PAST END OF TABLE?
23 057  140          JTC     NTFUND  /PAST END OF TABLE AND NO MATCH
        122
        023
23 062  006          LAI     ASCEND  /GET LOW LIMIT TO A FOR COMPARE
        305
23 064  276          CPL             /PAST END OF TABLE?
23 065  140          JTC     NTFUND  /PAST END OF TABLE AND NO MATCH
        122
        023
                /IF HERE, STILL IN TABLE.  TRY CONTENTS AGAIN.
23 070  301          LAB             /RETURN CHAR TO A REG
23 071  104          JMP     THISIT  /LOOP FOR NEXT TABLE ENTRY CHECK
        044
        023
23 074  006  CONVT,  LAI     MORTAB  /COMPUTE REL DISPLACEMENT LOW
        306
23 076  024          SUI     ASCTAB
        225
23 100  100          JFC     OK1
        104
        023
23 103  051          DCH             /HANDLE THE BORROW
23 104  206  OK1,    ADL             /ADD IN THE LOW POINTER
23 105  100          JFC     OK2
        111
        023
23 110  050          INH             /HANDLE THE CARRY
23 111  360  OK2,    LLA             /L IS NOW POINTING IN THE OUTPUT TABLE
23 112  006          LAI     MORTAB^ /COMPUTE RELATIVE DISPLACEMENT HIGH
        020
23 114  034          SBI     ASCTAB^
        020
23 116  205          ADH             /ADD IN THE HIGH POINTER
23 117  350          LHA             /H NOW POINTS IN THE OUTPUT TABLE
23 120  307          LAM             /REPLACEMENT CHAR TO A REG
23 121  007          RET             /CODE IN A REG RETURN
23 122  006  NTFUND, LAI     200     /CHAR NOT FOUND LOAD OUT A 200
        200
23 124  007          RET             /ERROR RETURN
```

```
                        /THIS SUBROUTINE TRANSLATES A REG TO PRINT MODE
23 125  074  XLAT,   CPI     377      /SET C FLG (GUARD BIT)
             377
23 127  022           RAL              /ROTATE
23 130  100           JFC     .-1      /IF NO LEFT GUARD, LOOP
             127
             023
                        /IF A REG CONTAINS A 000, ERROR CHAR WAS SEEN...
23 133  074           CPI     0        /SET FLGS
             000
23 135  110           JFZ     XLAT1    /NOT ERROR, TRANSLATE
             143
             023
23 140  006           LAI     DELSYM   /SET UP FOR DELETE SYMBOL
             134
23 142  007           RET              /EXIT
23 143  056  XLAT1,   LHI     MORTAB^  /POINT AT MORTAB
             020
23 145  066           LLI     MORTAB
             306
23 147  277  THIS,    CPM              /IS THIS THE CHAR?
23 150  150           JTZ     MCONVT   /YES, GO CONVERT THE CHAR
             177
             023
23 153  106           CAL     INCLH    /TRY NEXT CHAR
             046
             021
23 156  310           LBA              /SAVE CHAR IN B REG TEMP
23 157  006           LAI     MOREND^  /GET HIGH LIMIT TO A FOR COMPARE
             020
23 161  275           CPH              /PAST END OF TABLE?
23 162  140           JTC     NTFND    /YES, WITH NO MATCH
             225
             023
23 165  006           LAI     MOREND   /GET LOW LIMIT TO A FOR COMPARE
             366
23 167  276           CPL              /PAST END OF TABLE?
23 170  140           JTC     NTFND    /YES, NO MATCH
             225
             023
                        /IF HERE, STILL IN TABLE, TRY CONTENTS AGAIN
23 173  301           LAB              /RETURN CHAR TO A REG
23 174  104           JMP     THIS     /LOOP FOR NEXT TABLE ENTRY CHECK
             147
             023
23 177  006  MCONVT,  LAI     ASCTAB   /COMPUTE REL DISPLACEMENT LOW
             225
23 201  024           SUI     MORTAB
             306
23 203  140           JTC     MOK1
             207
             023
23 206  051           DCH              /HANDLE THE BORROW
23 207  206  MOK1,    ADL              /ADD IN THE LOW POINTER
23 210  100           JFC     MOK2
             214
             023
23 213  050           INH              /HANDLE THE CARRY
23 214  360  MOK2,    LLA              /L IS NOW POINTING IN OUTPUT TABLE
23 215  006           LAI     ASCTAB^  /COMPUTE RELATIVE DISPLACEMENT HIGH
             020
23 217  034           SBI     MORTAB^
             020
23 221  205           ADH              /ADD IN THE HIGH POINTER
23 222  350           LHA              /H NOW POINTS IN THE OUTPUT TABLE
23 223  307           LAM              /REPLACEMENT CHAR TO A REG
23 224  007           RET              /ASCII CODE IN A REG RETURN
23 225  006  NTFND,   LAI     BLANK    /CHAR NOT FOUND, LOAD OUT A SPACE
             040
23 227  007           RET              /ERROR RETURN




                        /SUBROUTINE TASK TO OUTPUT CODE
23 230  056  OTPUT,   LHI     OTFIFO^  /POINT AT STACK
             025
23 232  066           LLI     OTFIFO
             364
23 234  250           XRA              /CLEAR THE A REG
23 235  207           ADM              /CHAR COUNT TO A AND FLGS
23 236  053           RTZ              /NEXT TASK, IF NOTHING TO DO
23 237  106           CAL     INCLH    /POINT AT THE DATA
             046
             021
23 242  307           LAM              /CHARACTER TO A REG
23 243  074           CPI     DELETE   /ERROR CHARACTER?
             177
23 245  110           JFZ     OTPUT1   /NO, GO TRANSLATE
             264
             023
23 250  036           LDI     7        /YES, DO 8 DOTS
             007
23 252  106  OTERR,   CAL     DOT      /1 DOT
             120
             021
23 255  031           DCD              /-1 THE COUNT
23 256  110           JFZ     OTERR    /NOT DONE....DO IT AGAIN
             252
             023
23 261  104           JMP     OUTEND   /DONE, POP AND EXIT
             342
             023
23 264  106  OTPUT1,  CAL     XLATER   /TRANSLATE
             040
             023
23 267  330           LDA              /SAVE THE CHAR IN D REG
23 270  074           CPI     200      /IS IT A BAD CHAR (OR A SPACE)?
             200
23 272  110           JFZ     GOODCH   /CHAR OK, SO DO IT UP RIGHT
             311
             023
                        /IF HERE GENERATE A CHARACTER SPACE
23 275  036           LDI     6        /SET UP FOR 7 UNITS DELAY(UNITS-1)
             006
23 277  106  SPACE,   CAL     TICK     /WAIT ONE UNIT
             061
             021
23 302  031           DCD              /DECREMENT THE UNIT COUNTER
23 303  110           JFZ     SPACE    /LOOP UNTIL DONE
             277
             023
23 306  104           JMP     OUTEND
             342
             023
23 311  074  GOODCH,  CPI     200      /IF A=200 THEN DONE
             200
23 313  150           JTZ     OUTEND   /DONE!
             342
             023
23 316  100           JFC     DSH      /DOT OR DASH?
             327
             023
23 321  106           CAL     DOT      /YES A DOT, SO KEY OUT A DOT
             120
             021
23 324  104           JMP     OTLOOP   /NEXT SYMBOL
             332
             023
23 327  106  DSH,     CAL     DASH     /MUST BE A DASH, SO KEY A DASH
             126
             021
23 332  303  OTLOOP,  LAD              /GET THE CHAR BACK TO THE A REG
23 333  044           NDI     177      /THROW OUT THE USED BIT
             177
23 335  002           RLC              /ROTATE IN AN UNUSED BIT
23 336  330           LDA              /SAVE THE NEW IMAGE IN D
23 337  104           JMP     GOODCH   /LOOP FOR OTHER SYMBOLS
             311
             023
23 342  106  OUTEND,  CAL     TICK     /INTER LETTER SPACE
             061
             021
23 345  106           CAL     TICK
             061
             021
                        /SET UP TO POP THE STACK
23 350  066           LLI     OTFIFO   /POINT AT STACK
             364
23 352  056           LHI     OTFIFO^
             025
23 354  106           CAL     POP      /POP THE STACK
             014
             021
23 357  007           RET              /NEXT TASK
                        /END OF THE OUTPUT ENCODED DRIVE TASK SUBROUTINE




                        /KEYBOARD DECODER FOR NON COMMAND MODE
23 360  056  IDLE,    LHI     KYFIFO^  /POINT AT CHAR COUNT
             024
23 362  066           LLI     KYFIFO
             305
23 364  250           XRA              /CLEAR THE A REG
23 365  207           ADM              /CHAR COUNT TO A REG
23 366  053           RTZ              /BUFFER EMPTY, TRY SOMETHING ELSE
23 367  106           CAL     INCLH    /POINT AT CHAR
             046
             021
23 372  347           LEM              /SAVE CHAR IN E REG TEMP
23 373  106           CAL     DCRLH    /POINT AT START OF BUFFER
             052
             021
23 376  106           CAL     POP      /POP THE CHAR OFF THE BUFFER
             014
             021
24 001  304           LAE              /CHAR TO A REG
24 002  074           CPI     ESC      /IS IT AN ESC?
             175
24 004  150           JTZ     IDLE1    /YES
             025
             024
24 007  056           LHI     OTFIFO^  /OUTPUT IN CODE
             025
24 011  066           LLI     OTFIFO   /SET UP FOR ENTPAK
             364
24 013  310           LBA              /DATA IN B REG
24 014  026           LCI     BUFOUT   /SIZE IN C REG
             060
24 016  106           CAL     ENTPAK
             367
             020
24 021  112           CFZ     WHOOP    /BUFFER FULL, TELL USER
             305
             021
24 024  007           RET              /DONE
24 025  056  IDLE1,   LHI     CMMND^   /SET FOR COMMAND MODE
             024
24 027  066           LLI     CMMND
             275
24 031  076           LMI     1        /MODE=1
             001
24 033  007           RET




                        /SUBROUTINE TO SERVICE MORSE CODE INPUT
24 034  105  INPEND,  SENSE            /GET CODE INPUT LINE
24 035  044           NDI     1        /WE USE THE LSB
             001
24 037  013           RFZ              /NOTHING PENDING, EXIT
24 040  056           LHI     INCHAR^  /POINT AT HOLDING REG
             024
24 042  066           LLI     INCHAR
             304
24 044  076           LMI     1        /SET UP TO SHIFT IN MORSE
             001
24 046  056  INTIME,  LHI     TIMER^   /POINT AT TIMER REG
             024
24 050  066           LLI     TIMER
             303
24 052  076           LMI     0        /INITIALIZE FOR TIME=0
             000
24 054  106  INSENS,  CAL     TICKI    /WAIT FOR PART OF A BAUD (1/8)
             107
             021
24 057  056           LHI     TIMER^   /UPDATE TIMER
             024
24 061  066           LLI     TIMER
             303
24 063  317           LBM
24 064  010           INB              /+1
24 065  371           LMB              /TIMER+1
24 066  105           SENSE            /KEY DOWN?
24 067  044           NDI     1
             001
24 071  150           JTZ     INSENS   /WAIT FOR KEY UP
             054
             024
                        /IF HERE, KEY IS NOW UP
24 074  046           LEI     0        /SET E=0 FOR DOT, FIX LATER IF DASH
             000
```

```
24 076  301          LAB             /TIMER TO A REG (B** OF BAUD)
24 077  074          CPI     20
        020
24 101  100          JFC     INDASH  /IF DASH, SERVICE DASH
        131
        024
                     /SEE IF CLOCK MUCH TOO SLOW FOR DOT
24 104  074          CPI     6       /SHOULD BE A 10 IDEAL
        006
24 106  100          JFC     INPOK   /CLOCK IS GOOD ENOUGH
        156
        024
24 111  056          LHI     BAUDI^  /NOT GOOD ENOUGH, FIX WPM CONSTANT
        024
24 113  066          LLI     BAUDI
        302
24 115  307          LAM             /WPM TO A
24 116  074          CPI     2       /WPM TOO LOW TO TRACK?
        002
24 120  140          JTC     INPOK   /DON'T TRY TO FIX, ALREADY TOO FAST.
        156
        024
24 123  034          SBI     1       /-1
        001
24 125  370          LMA             /BAUDI-1
24 126  104          JMP     INPOK
        156
        024

                     /IF HERE, SYMBOL IS DASH, UPDATE BAUDI FOR TRACKING
24 131  056  INDASH, LHI     BAUDI^  /POINT AT INPUT WPM
        024
24 133  066          LLI     BAUDI
        302
24 135  301          LAB             /GET TIMER TO A REG AGAIN
24 136  074          CPI     34      /CLOCK TOO FAST?
        034
24 140  140          JTC     OKDASH  /NO.
        154
        024
24 143  307          LAM             /YES
24 144  074          CPI     376     /TIMER REALLY TOO TOO SLOW?
        376
24 146  100          JFC     OKDASH  /YES, BAIL OUT!
        154
        024
24 151  004          ADI     1       /+1
        001
24 153  370          LMA             /BAUDI+1
24 154  046  OKDASH, LEI     1       /SET E=1 FOR DASH
        001
24 156  056  INPOK,  LHI     INCHAR^ /POINT AT CHAR HOLDING REG
        024
24 160  066          LLI     INCHAR
        304
24 162  307          LAM             /GET PARTIAL CHAR TO A REG
24 163  022          RAL             /SHIFT UP ONE BIT
24 164  044          NDI     376     /JUNK THE OLD CARRY BIT
        376
24 166  264          ORE             /BRING IN NEW SYMBOL FROM E REG
24 167  370          LMA             /NEW PARTIAL CHAR TO INCHAR

                     /TIME THE INTERSPACE TO FIND WHAT TYPE IT IS.
24 170  056          LHI     TIMER^  /RESET THE TIMER
        024
24 172  066          LLI     TIMER
        303
24 174  076          LMI     0       /TIMER RESET
        000
24 176  106  UPTIME, CAL     TICKI   /DELAY 1/8 OF A BAUD TIME
        107
        021
24 201  105          SENSE           /GET THE KEY STATUS
24 202  044          NDI     1
        001
24 204  150          JTZ     INTIME  /KEY DOWN, GET NEXT SYMBOL
        046
        024
24 207  056          LHI     TIMER^  /UPDATE THE TIME
        024
24 211  066          LLI     TIMER
        303
24 213  317          LBM
24 214  010          INB             /+1
24 215  371          LMB             /TIMER+1
24 216  301          LAB             /GET TIMER TO A FOR COMPARE
24 217  074          CPI     377     /END OF MESSAGE?
        377
24 221  110          JFZ     NOTEOM  /KEEP LOOPING,
        235
        024
24 224  016          LBI     LF      /END OF MESSAGE, CR-LF-CR-LF
        012
24 226  106          CAL     PPAK
        304
        022
24 231  106          CAL     PPAK
        304
        022
24 234  007          RET             /EXIT TO MAINLINE
24 235  074  NOTEOM, CPI     60      /END OF WORD?
        060
24 237  110          JFZ     ENDLET  /NO.
        247
        024
24 242  006          LAI     1       /YES, OUTPUT A SPACE
        001
24 244  104          JMP     MPAK    /DO IT OUT RIGHT.
        263
        024
24 247  074  ENDLET, CPI     24      /END OF LETTER?
        024
24 251  110          JFZ     UPTIME  /NO, KEEP TIMING THE UP TIME
        176
        024
24 254  056          LHI     INCHAR^ /POINT AT HOLDING REG
        024
24 256  066          LLI     INCHAR
        304
24 260  307          LAM             /MORSE FROM HOLDING REG TO A REG
24 261  076          LMI     1       /RESET THE HOLDING REG FOR NEXT CHAR
        001
24 263  106  MPAK,   CAL     XLAT    /ASCII TO A EQUIV OF MORSE
        125
        023

24 266  310          LBA             /SET UP FOR PPAK
24 267  106          CAL     PPAK    /PRINT THE CHARACTER
        304
        022
24 272  104          JMP     UPTIME  /KEEP TIMING THE UP TIME
        176
        024

                     /***ANYTHING BEFORE THIS POINT CAN BE IN PROM***
                     /***EVERYTHING AFTER THIS POINT MUST BE IN RAM***

24 275  000  CMMND,  DATA    0       /0=NORMAL MODE, OTHERWISE COMMAND MODE
24 276  000  TWIDTH, DATA    0       /WHEN BYTE IS ZERO, GENERATE A CR/LF
24 277  000  MSSCNT, DATA    0       /TEMP CHARACTER COUNT FOR MESSAGE DUMP
24 300  000  SOH,    DATA    0       /SUBMODE FOR DMPSUB   0=PRINT   1=SEND
24 301  000  BAUD,   DATA    0       /WPM CONSTANT (SEE HEADING ON PROGRAM)
24 302  000  BAUDI,  DATA    0       /INPUT WPM VALUE (GETS MODIFIED)
24 303  000  TIMER,  DATA    0       /TIME BAUD *8 (10 OCTAL) COUNTER
24 304  000  INCHAR, DATA    0       /INPUT CHAR HOLDING REG

24 305  000  KYFIFO, HLT             /INPUT BUFFER

                     *KYFIFO+BUFSKY
25 304  000  PNFIFO, HLT             /PRINTER BUFFER

                     *PNFIFO+BUFSPN
25 364  000  OTFIFO, HLT             /OUTPUT BUFFER FOR CODE

                     *OTFIFO+BUFOUT
26 044  000  MSSGBF, HLT             /MESSAGE BUFFER
```

```
ASCEND   20 305
ASCTAB   20 225
BAUD     24 301
BAUDI    24 302
BLANK    00 040
BUFOUT   00 060
BUFSKY   00 377
BUFSPN   00 060
C        00 103
CLRMD    22 153
CMMND    24 275
CMMNDR   21 150
CONVT    23 074
CR       00 015
DASH     21 126
DCRLH    21 052
DEL      23 034
DELETE   00 177
DELSYM   00 134
DMPSUB   21 333
DOT      21 120
DSH      23 327
ENDLET   24 247
ENTPAK   20 367
ERCHAR   00 007
ERROFL   21 011
ESC      00 175
ESCSYM   00 044
ETX      00 003
FINDOT   21 137
GOODCH   23 311
IDLE     23 360
IDLE1    24 025
INCHAR   24 304
INCLH    21 046
INCRLF   23 031
INDASH   24 131
INPEND   24 034
INPOK    24 156
INSENS   24 054
INTIME   24 046
KYBD     22 227
KYFIFO   24 305
L        00 114
LDNXT    21 224
LDNXT1   21 232
LDNXT2   21 272
LF       00 012
MCONVT   23 177
MOK1     23 207
MOK2     23 214
MOREND   20 366
MORTAB   20 306
MPAK     24 263
MSGSZ    00 377
MSSCNT   24 277
MSSGBF   26 044
NETX     22 267
```

```
NOTEOM   24 235
NTFND    23 225
NTFUND   23 122
NXTPNT   22 361
OK       21 006
OKDASH   24 154
OK1      23 104
OK2      23 111
OTERR    23 252
OTFIFO   25 364
OTLOOP   23 332
OTPUT    23 230
OTPUT1   23 264
OUTEND   23 342
P        00 120
PNFIFO   25 304
PNTR     22 316
PNT1     23 027
POP      21 014
POPLOP   21 023
POPY     21 036
PPAK     22 304
PRT      21 317
PRT1     21 341
PRT2     22 347
PR2      22 004
PR3      22 035
PR4      21 372
QUEST    00 077
RESTRT   20 172
S        00 123
SNDNX    22 060
SOH      24 300
SOH1     22 024
SPACE    23 277
STRT     20 120
STRT1    20 140
T        00 124
TEST     22 074
TEST1    22 120
THIS     23 147
THISIT   23 044
TICK     21 061
TICKI    21 107
TIMER    24 303
TWIDTH   24 276
UNPAK    22 164
UPARRO   00 136
UPTIME   24 176
W        00 127
WAITMT   22 220
WAIT1    21 070
WAIT2    21 067
WHOOP    21 305
WIDTH    00 110
WPM      22 133
XLAT     23 125
XLATER   23 040
XLAT1    23 143
```

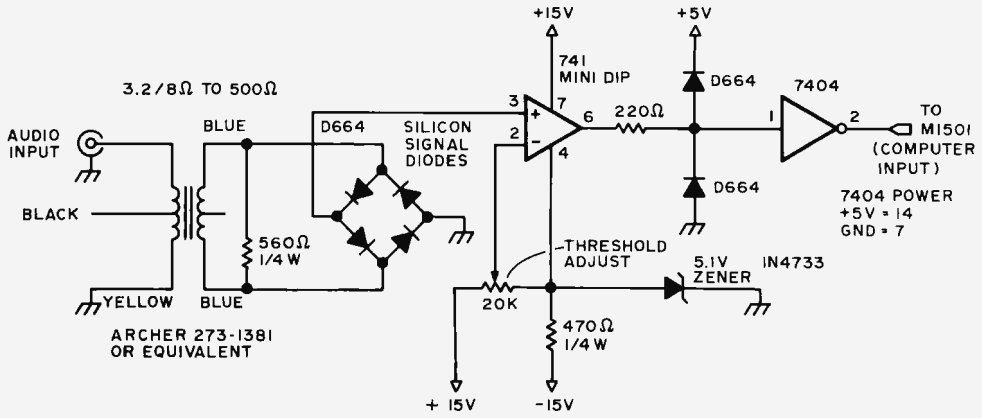


Figure 5: The input circuit used by the author in developing the Morse code interpreter. This circuit works but is not optimal. The operational amplifier saturates if any AC signal is present on the input to the bridge rectifier, so the frequency selectivity of this circuit is virtually non-existent (ie: all stations heard in the pass band of your receiver will be logically "ORed" leading to garbled copy if you operate in a crowded band).

Text continued from page 56

used since they are all driven in the same sequence). In both of the above cases the keying sense for input and output is a TTL low level for a "key down" condition ("mark") and a TTL high level for a "key up" condition ("space").

This program has been tested on a starter set and has successfully operated in both PROM and programmable memory. When used in programmable memory, it should be noted that a DEC M7344YB (or an extra 1 K of programmable memory over the starter set M7344YA) is the minimum memory requirement. When the program is assembled and programmed into PROM, approximately 4 1/2 EROMs (1702A) are required.

Experience to Date

The program has been tested in generating Morse code over the speed range of 7.2 wpm to 120 wpm and appears to function properly. The program has been tested in receiving Morse code over the speed range of 7.2 wpm to 96 wpm: up to about 63 wpm the decoding function is fairly acceptable; at 89 wpm the number of erroneous characters is considered to be unacceptable by the author in this particular test. It is the author's opinion that the error rate at the higher code input speeds is probably related to the design of a particular input processing circuit that was used (see figure 5). In general, a phase locked loop, or a similar highly selective decoding scheme, would be useful, particularly to an amateur radio operator working the crowded bands of a field day type event. One such circuit is illustrated in figure 6. This would provide the amateur with a printout of communications in both directions from the station operating. In fact, with sufficient comment being transmitted to and from the stations involved, the printout from a hard copy terminal would provide a log for the field day events.■

Some Thoughts on Improvements and Adaptations

The Morse code interpreter described here has been implemented and used by the author. As in any design, there is room for improvement and expansions of the capacity of the program. Here are some suggestions:

- Design a good input filter to pick up audio and output digital (phase locked loop?). See figure 6.
- Add multiple message buffers.
- Use multiple precision arithmetic to provide a larger message buffer.
- Modify the program for RTTY (Replace Morse table with Baudot table, add single byte flag to keep track of FIGS vs LTRS modes).
- Wire 6.3 VAC at 60 Hz into the DEC M7346 module and write a real time clock routine to keep track of time of day.

For individuals with 8080 processors, or the new Z-80, the source code for this Morse code interpreter (see listing 2) can be translated on a one to one basis into code for these newer computers. Such code will work without major changes, but will not make optimal use of the expanded instruction sets.

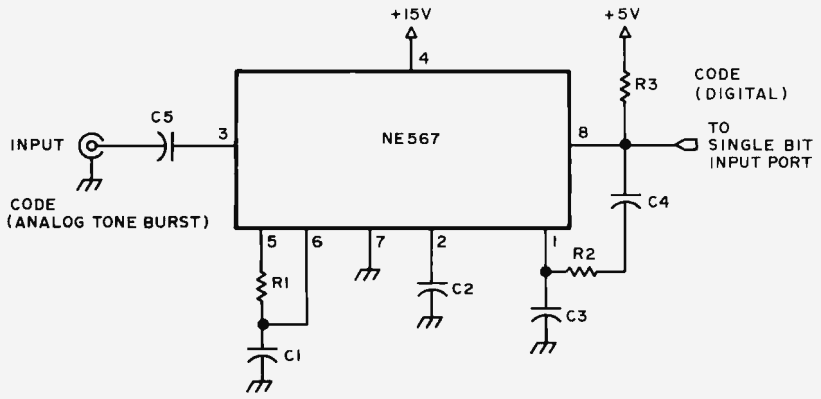


Figure 6: A suggested selective input filter for better performance. This circuit is adapted from the Signetics Catalog, page 6-97. The design equations are shown. The phase lock loop "latches up" when the signal is detected in the filter's band, typically after several tens of cycles. C3 determines the time taken for the filter to detect this condition. The time taken to unlatch after the signal disappears is determined by R2 and C4, with some effects from C3.