MORSE CODE
COMMUNICATION AID
FOR
THE HANDICAPPED
MORSE CODE
COMMUNICATION AID
FOR
THE HANDICAPPED

by

E. G. CALLWAY, B. ENG.

A Project
Submitted to the School of Graduate Studies
in Partial Fulfillment of the Requirements for the Degree
Master of Engineering

McMaster University
April 1981
TITLE: Morse Code Communication Aid for the Handicapped

AUTHOR: E.G. Callway, B.Eng.

SUPERVISOR: Dr. Ralph Bloch

NO. OF PAGES 122, vi
ABSTRACT

A microprocessor driven display was built and programmed for the storage and reproduction of Bliss symbols. An explanation is offered for the success of the symbol language in teaching the handicapped.

The hardware was designed to be inexpensive enough for classroom use, but still deliver adequate flexibility and resolution. Due to the complexity and variety of the symbols a method of data compaction was developed to reduce the required storage space.

Initial tests are presented and suggestions are made for continuing the work.
ACKNOWLEDGEMENTS

The author would like to thank Dr. R. Bloch, Chedoke Hospital, for his advice and guidance. Appreciation is also expressed to Ted Bojanowski, Madeleine Harris, Ted Iler and Tony Wallace for their continual advice and support.

Special thanks are due to Dr. J. Russell for the interest he took in the project.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>CONTENT</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.0</td>
<td>Communication Aids for the Handicapped</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>The Morse Code as a Communication Aid</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Previous Work and Basic Requirements</td>
<td>2</td>
</tr>
<tr>
<td>II</td>
<td>THE MORSE CODE</td>
<td>4</td>
</tr>
<tr>
<td>2.0</td>
<td>Basic Description</td>
<td>4</td>
</tr>
<tr>
<td>2.1</td>
<td>Code Sample</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>Decoding</td>
<td>6</td>
</tr>
<tr>
<td>2.3</td>
<td>Code Characterization Displays</td>
<td>8</td>
</tr>
<tr>
<td>2.4</td>
<td>Decoder Functions</td>
<td>9</td>
</tr>
<tr>
<td>2.5</td>
<td>General Decision Strategies</td>
<td>13</td>
</tr>
<tr>
<td>2.6</td>
<td>Decision Problems and Approaches</td>
<td>14</td>
</tr>
<tr>
<td>III</td>
<td>THE DECODING PROGRAM</td>
<td>20</td>
</tr>
<tr>
<td>3.0</td>
<td>Timing</td>
<td>20</td>
</tr>
<tr>
<td>3.1</td>
<td>Input</td>
<td>20</td>
</tr>
<tr>
<td>3.2</td>
<td>Histograms</td>
<td>21</td>
</tr>
<tr>
<td>3.3</td>
<td>Morse/ASCII Conversion</td>
<td>21</td>
</tr>
<tr>
<td>3.4</td>
<td>Slow Adaptability</td>
<td>22</td>
</tr>
<tr>
<td>3.5</td>
<td>Threshold Setting</td>
<td>22</td>
</tr>
<tr>
<td>3.6</td>
<td>Out of Range Correction</td>
<td>28</td>
</tr>
<tr>
<td>IV</td>
<td>HARDWARE - AIMS AND DEVELOPMENT</td>
<td>30</td>
</tr>
<tr>
<td>4.0</td>
<td>Hardware Requirements</td>
<td>30</td>
</tr>
<tr>
<td>4.1</td>
<td>Patient Interface</td>
<td>30</td>
</tr>
<tr>
<td>4.2</td>
<td>Automatic Keyer</td>
<td>35</td>
</tr>
<tr>
<td>4.3</td>
<td>Detailed Keyer Operation</td>
<td>37</td>
</tr>
<tr>
<td>4.4</td>
<td>Footswitch/Keyer Results</td>
<td>40</td>
</tr>
<tr>
<td>4.5</td>
<td>Audio Input Board</td>
<td>40</td>
</tr>
<tr>
<td>4.6</td>
<td>Digital Input Board</td>
<td>46</td>
</tr>
<tr>
<td>4.7</td>
<td>Microprocessor System</td>
<td>48</td>
</tr>
<tr>
<td>V</td>
<td>RESULTS AND CONCLUSIONS</td>
<td>51</td>
</tr>
<tr>
<td>5.0</td>
<td>Testing and Results</td>
<td>51</td>
</tr>
<tr>
<td>5.1</td>
<td>Trial Unit</td>
<td>53</td>
</tr>
<tr>
<td>5.2</td>
<td>Conclusions and Recommendations</td>
<td>53</td>
</tr>
</tbody>
</table>
## APPENDICES

| A | Morse Code Characters and Program Commands | 56 |
| B | Flowcharts | 58 |
| C | Monitor Program | 82 |
| D | Morse Decoding Program | 98 |

## REFERENCES

| REFERENCES | 121 |
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Histograms</td>
<td>7</td>
</tr>
<tr>
<td>2. Scatter Plot</td>
<td>10</td>
</tr>
<tr>
<td>3. Teletype Histogram</td>
<td>11</td>
</tr>
<tr>
<td>4. Tektronix Histogram</td>
<td>12</td>
</tr>
<tr>
<td>5. Threshold Histograms</td>
<td>23</td>
</tr>
<tr>
<td>6. Rancho Footswitch</td>
<td>33</td>
</tr>
<tr>
<td>7. Metal Footswitch</td>
<td>33</td>
</tr>
<tr>
<td>8. Wooden Footswitch</td>
<td>36</td>
</tr>
<tr>
<td>9. Keyer Schematic</td>
<td>38</td>
</tr>
<tr>
<td>10. Final Footswitch</td>
<td>41</td>
</tr>
<tr>
<td>11. Audio Input Schematic</td>
<td>42</td>
</tr>
<tr>
<td>12. PLL Tests</td>
<td>45</td>
</tr>
<tr>
<td>13. Digital Input Schematic</td>
<td>47</td>
</tr>
<tr>
<td>14. Development System</td>
<td>49</td>
</tr>
<tr>
<td>15. Trial Unit</td>
<td>54</td>
</tr>
</tbody>
</table>
CHAPTER I

INTRODUCTION

1.0 Communication Aids for the Handicapped

The inability to communicate with other people can make life difficult and unrewarding. Since the most common form of communication is speech, lack of speech can be seen as a serious handicap.

Communication aids enable a person with a speech, visual, auditory or motor disorder to communicate with other people. Those with handicaps in these areas are often downgraded, as a lack of words is unfairly equated with a lack of ideas.

The purpose of this project was to provide a non-vocal cerebral palsied person with a portable means of communication using Morse code. An appropriate review of cerebral palsy can be found in Day (4). Briefly, it is a neurological disorder, present at birth and nonincreasing, which causes widespread motor disabilities. Its effects vary from slight tremors and weakness to a complete lack of control over voluntary movements, including speech.

In this case, the subject was a young man confined to a wheelchair and possessing extremely limited speech, but of
normal intelligence. He had some control over his feet and was able to turn the pages of a book and type slowly with them. For practical reasons of size and weight a typewriter could not be permanently attached to his wheelchair. An electronic keyboard with an eye-level display might have worked, but it would have still required a ruggedized QWERTY keyboard small enough to be unobtrusive, but large enough to be worked with the feet. For use outdoors either unit would be unsuitable as they could not be operated with a shoe.

1.1 The Morse Code as a Communication Aid

It was decided to try the Morse code as a method of communication. Unlike a typewriter, a single switch can be used to send the whole code, thus solving the problems of size and ruggedness. Morse code is fairly compact, with common letters such as E or T being a single dot or dash, while Z is much longer. This fortuitous arrangement is only valid for English.

The main disadvantage is that most people do not understand the code. It was felt that a simple and reliable decoding device could be built using microprocessor technology.

1.2 Previous Work and Basic Requirements

Schemes for the automatic decoding of Morse code have always been popular, but not necessarily cheap or effective.
Most of the research in this area is performed by amateur radio operators and various military organizations, each for their own purposes. Bell (2) presents the military view with its aim of increased surveillance with less manpower. For ham radio operators it becomes possible to communicate more enjoyably by removing some of the hard work. Recently there has been work on microprocessor based decoders, but they tend to require large volumes of code for initialization (1) or require fairly good code (12). The first requirement cannot be met due to the short conversations, and the second because of the nature of the sender. It was felt that a better algorithm could be developed for this use.

Unlike receiving, sending the code is fairly easy, and in fact the subject learned to send it within one week. By contrast, this author still cannot understand the code without the aid of the device developed here!

The items generated by this project were a footswitch interface to send the code, an adaptive software program to interpret it, and recommendations for the hardware required to produce a portable unit.
CHAPTER II

THE MORSE CODE

2.0 Basic Description

Morse code is an internationally recognized binary (on-off) coding scheme for the transmission of letters and numbers. Although it was originally designed to be sent and received by human operators, it can also be used with mechanical or electronic devices.

Each character consists of a unique sequence of one to six MARKS (on) separated by SPACES (off). The marks and spaces, or ELEMENTS, have several different lengths. Marks of time duration 1 are called DOTS, and those of lengths 3 are DASHES, SYMBOL SPACES of length 1 are used inside a letter, CHARACTER SPACES of length 3 separate characters, and WORD SPACES of duration 7 divide words.

The absolute timing of the elements does not matter as long as they maintain the 1:3:7 ratios. The dot is usually designated as the speed determining element. By not fixing the speed, operators can work from 5 to 60 words per minute (Guenther, 9), often limited by skill of the receiver. Machine to machine transfers can proceed at higher rates.

Grappel and Hemenway (8) state that the reciprocal of
the speed in words per minute corresponds roughly to the 
dot length in seconds for normal text. Thus, at an average 
speed of 10 words per minute, the dots and symbol spaces 
would be 0.1 second long.

Bell (2) was unable to find a standard value for the 
bandwidth of Morse code, but used 3 times the reciprocal of 
the dot length as a working value. This figure is important 
when designing analog input circuitry to reduce noise while 
retaining information. The bandwidth of machine code could 
be rigorously calculated, but hand sent code is too depen­
dent on human factors for meaningful results.

2.1 Code Sample

To send a message, such as "I am hot", it is neces­
sary to find the correct code for each letter, and then 
join the individual codes with the correct spaces. A table 
of the various character codes is given in Appendix A. 
There is no provision for upper or lower case in the code.

Individual codes:

I . . a . - m - -
h . . . o - - t -

Complete message:

I a m h o t (text) (code)
1 1 1 1 3 3 3 3 3 3 (mark times)
1 7 1 3 7 1 1 1 3 1 1 3 (space times)
It can be seen that information is contained in both the marks and spaces.

2.2 Decoding

To decode a message it is first necessary to take each element and decide its relative length: 1, 3 or 7. For well sent, low noise code, the wide ratios (1:3 and 3:7) make this an easy task, even for a mediocre operator or simple machine.

Because the spaces clearly define the beginning and end of letters and words, it merely remains to look up the individual letter codes and produce a copy of the message.

The above tasks can be implemented easily to produce a simple decoder at low cost. The Morse-A-Letter (11) is an example of such a device, available as a kit for $150.

Unfortunately, Morse code is rarely received under ideal conditions. Human operators cannot exactly duplicate the 1:3:7 ratios, and during a long message both the ratios and overall speed will vary. Bell (2), Guenther (9), Freimar (5) and other detail some of these changes, including the variations which may occur inside a single character.

For most applications the signal has been received via a radio link, and is further corrupted by noise and fading.

For reliable decoding of Morse code, the machine must be adaptive in some way. It may only adjust its overall
a) Bell, Reference 2.

b) Hickey, Reference 10.

Figure 1. Histograms
speed, such as the Automatic Fist Follower (7), it may com-
pensate for deviations in the 1:3:7 ratios (3), or it may
even perform some textual analysis based on a rudimentary
knowledge of English (10).

2.3 Code Characterization Displays

In order to analyze a Morse signal, most researchers
have found it helpful to have a graphical representation of
the main characteristics.

Hickey (10) and Bell (2) used histograms, shown in
Figure 1. The horizontal axis is time, and the positive
vertical axis shows the number of marks with that duration.
The negative vertical axis displays the space information,
inverted for clarity. In both cases, some of the important
characteristics of Morse code are shown:

1. There are more dots than dashes, and more symbol spaces
   than character or word spaces.
2. The dot cluster is narrow with an obvious midpoint.
3. The dash cluster is wide, and the midpoint may not be
   clearly defined (flat top, rather than peak).
4. The symbol space cluster is as well-defined as the dot
   cluster, with nearly the same midpoint.
5. The character spaces are poorly grouped.
6. The word spaces are not grouped at all.

Point 1 is inherent in any Morse transmission of text,
but 2 to 6 are peculiar to hand sent code. It can be seen that the 1:3:7 ratios are not exactly followed.

Guenther (9) used scatter distribution plots which are able to show the changes when one element follows another. For the example shown in Figure 2, it can be seen that dashes following word or character spaces (groups D and E) are longer than dashes after symbol spaces (group F). This particular type of information cannot be easily seen on the histogram displays. The frequency of occurrence information is slightly obscured by the scatter plots as it changes from height to dot density.

For this project, the histogram method was chosen as it offers a good display of information with minimal hardware and software requirements. The output was first presented on a Teletype printer (Figure 3), but the program was changed to use a Tektronix graphics terminal (Figure 4). This gave a higher quality display and much faster output.

2.4 Decoder Functions

Any decoding device must perform several basic functions:

1. Separate the marks into dots and dashes.

This is the least difficult operation as the dots and dashes are usually well sent, preserving the 1:3 ratio or even making it larger (3).
Figure 2. Scatter Plot. (Guenther, Ref. 9)
Figure 3: Teletype Histogram

0 1 2 3 4 5 6 7 8 9
A B C D E F

0 1 2 3 4 5 6 7 8 9
A B C D E F

0 1 2 3 4 5 6 7 8 9
A B C D E F
Fig. 4  Annotated Tektronix Histogram
2. Separate the spaces into symbol, character and word. The spaces are never sent with as much care as the marks. Blair (3) and Guenther (9) agree that a more complicated decision process is needed for the spaces.

3. Based on the character spaces which have been found, divide the data stream into characters, decode and output them.

If the character spaces have been correctly found, and if the marks were properly identified, the final conversion from Morse to text is a simple operation using a look-up table. When the above processes produce an invalid character, there is a choice between printing an error message, and reassigning the various elements using a modified decision technique in an attempt to find a valid character. This correction process may be quite involved in the larger devices (11).

4. Based on the word spaces, output spaces between words.

Although word spaces are the most variable type of space, their identification is not critical, as a missing space between words rarely causes a loss of understanding.

2.5 General Decision Strategies

Hickey (10) noted that there are three basic approaches to the manual Morse problem:

1. Macro: You can accumulate statistical information on an
operator and use this data to make decoding decisions.

2. Micro: You can make your decisions on a mark to mark basis, sometimes called the "idiot dot" method.

3. Hybrid: You can compromise these two methods and come up with a hybrid algorithm.

Method 1 requires a memory large enough to contain all the timing information in a message, but should give the best results. Method 2 requires very little computer time or space, but is the least reliable. Successful devices, such as MAUDE (6), use the hybrid approach.

A common method is to keep an average speed parameter which varies slowly, and an average dot time which uses only the last few characters. The decision thresholds are calculated from empirical formulas using the dot average. The short "time constant" on the dot average allows tracking of fast speed changes, while the slower responding speed parameter inhibits locking on t. noise (8).

2.6 Decision Problems and Approaches

1. Decoding Delays:

The methods which build up a large body statistical information about an operator require a large sample of code to be effective, thus delaying decoding. Guenther (9) tried to quicken the process by decoding initially with one set of rules, then switching to a better set when more code
had been received.

For use as a conversational communication aid, the long initialization times of most methods are unacceptable. Since the code will be sent very slowly (possibly 1 w.p.m.) it is important to have each letter decoded as quickly as possible.

2. Non-Code Activation:

Due to the nature of the sender, there are periods when the code switch is involuntarily activated. If some of the simpler averaging techniques are used, after several non-code "words", the program will have locked on to that style, losing its memory of the sender's true characteristics. This is similar to the problems caused by noise and interference in the radio case, and is why "idiot dot" strategies fail.

Since a non-code sequence is obvious to an observer, (because of the sender's physical involvement) no attempt was made to block decoding during one. It would require a complex program to have the computer recognize invalid input. The decoding algorithm was designed so that inputs which are not similar to previously received code do not affect the decoding thresholds unless prolonged.

3. Space Problems:

Problems arise when there is poor distinction between the symbol and character spaces. If a long group of marks
is received with no obvious character space in the string, a good receiver can separate the string into characters using his knowledge of valid letter codes, combined with his expectations based on context.

Machines find this difficult unless they contain rules and vocabulary from the language being decoded. An easier method is to mix timing and simple language rules such as in the MAUDE decoder. Rules such as "The longest of 6 successive spaces is almost always a character space" allowed the machine to divide a string of poor code into letters. If the division resulted in an invalid character, a modified set of rules was tried. (6)

The bad grouping of the various space types not only causes incorrect decoding (due purely to the bad sending), but also makes it difficult to calculate best values for the space thresholds, compounding the errors. In this program the threshold setting process does disregard poorly grouped information as mentioned above, but in addition the initially calculated symbol/character threshold is averaged with the dot/dash threshold, which is more reliable. For perfectly sent code these two thresholds should be equal.

4. Correction Routines:

McElwain and Evens (11) reported good success with a "degarbling" program to correct machine received code. It was given a vocabulary list of every word which might be
used in the message, along with the received, partially
decoded text, and some timing information. Based on this
it could correct many of the errors in the received text.
This would be useful in a military situation where the con­
text is unknown but the vocabulary prescribed. It allows
large volumes of readable text to be produced with little
operator attention.

In ordinary conversation the vocabulary is large, but
the context is known - such as replies to questions. It was
found in preliminary tests that a reply never had to be
completely spelled out, as the listener could correctly
guess the full statement after a few letters or words. For
this reason no error correction routines were included in
this design. This works well in a conversational environ­
ment, but might be inadequate for other uses.

5. Effects of Incorrect Decisions:

Most of the schemes examined use variations of the
following method. The element being processed (on line
or from memory) is examined and a decision made as to its
nature (dot/dash, symbol/character/word). This decision is
used as part of the character currently being assembled for
decoding. If the element was classified incorrectly, the
current character will be incorrect. Error correction
schemes may help if an invalid character was produced.
One incorrect character is not a problem, but each element is then used to update a running average, as in:

$$\text{DOT AVG.} = \text{DOT AVG.} + \frac{\text{NEW DOT}}{8} - \frac{\text{DOT AVG.}}{8}$$

$$\text{DASH AVG.} = \text{DASH AVG.} + \frac{\text{NEW DASH}}{8} - \frac{\text{DASH AVG.}}{8}$$

(This keeps running averages of the typical dot and dash, with each new element given a weight of 0.125.)

The various averages are used to calculate the thresholds, as in:

$$\text{MARK AVG.} = \frac{\text{DOT AVG.}}{4} + \frac{\text{DASH AVG.}}{2}$$

(For perfectly weighted 1:3 code, this produces a threshold at 1.75, an empirically better value than value 2, which might have been expected.) (after 1)

Unfortunately, any incorrectly classified element is averaged into the wrong place.

The effect of this is that a period of bad code or noise is not only incorrectly decoded (which may be unavoidable), but is also used to calculate incorrect thresholds which prolong decoding errors into periods of good code.

Most of the element classification errors occur when an element is received with a timing value near a threshold value. These elements are poorly grouped with respect to the average of their intended value. If these elements are included in the average calculations, even if they are correctly classified, poor threshold values must result.
Elements near present threshold values cannot however be explicitly and permanently excluded from calculations as they may be important if the operator has changed speed.

The decoding algorithm outlined in the next chapter tries to use only well-grouped data, ignoring data which may represent noise or odd code.
CHAPTER III

THE DECODING PROGRAM

Some of the main features of this program are:

3.0 **Timing**

A software timer is used instead of external counters. This gives maximum adaptability and minimizes the hardware requirements. At the slow code speeds anticipated, the non-timing portions of the program do not significantly distort the timing, so the delays through various paths were not equalized, but ignored. A single speed parameter is used, with an estimated useful range of 300 to 1.

All timing in the program is in the arbitrary, dynamically variable units determined by the speed parameter.

3.1 **Input**

The code input is sampled under program control, instead of using an interrupt. It was felt that an interrupt based system was too susceptible to noise, as it must respond to every input change, however short. When fully adapted to a code style, an average length dot is sampled 8 times, a dash 24 times. While adapting, the program will decode correctly if a dot is sampled from 2 to 20 times.
The program will lock on to code over a much wider range.

3.2 **Histograms**

As each mark or space finishes, its length (in arbitrary timing parameter units) is stored in a histogram type table. There are separate tables for marks and spaces. The tables allow for a length from 1 to 64 timing units, with a maximum count in each entry bin of 64 occurrences. When any bin in either the mark or space table reaches 64 counts, the data in both tables is divided by 2. This prevents software overflows and allows new data to slowly take over from the old. The histograms are smoothed with a moving window and used for threshold calculations.

The smoothed data can be displayed on a graphics terminal as an actual histogram. This is useful for checking the overall quality of the code and the validity of the threshold setting algorithm.

3.3 **Morse/ASCII Conversion**

As each element comes in, it is shifted into a character holding register, using "0" for dots and "1" for dashes. When a character space is found, the data in the character holding register is used directly as a memory address pointing to the equivalent ASCII in a look-up table. This method is simple and fast while requiring a minimum of space for the table.
Two 6 mark characters produce ambiguous addresses and are dealt with separately.

If it is desired to put out more than one character per Morse input, such as a "CR/LF", then a special case subroutine must be written.

3.4. **Slow Adaptability**

After the thresholding process has decided dot or dash each mark time is compared to 8 or 24 respectively. If the time was short, the overall speed is increased by a small increment, and vice versa. This action slowly forces the program into the correct speed range. As it works slowly, noise or bad code will not really affect the speed unless prolonged for dozens of marks, an unlikely event. Due to its poor quality, the space data is not used.

Initially this was the only adaptation mechanism in the program, but it could not be made fast-responding without the program becoming unstable during bad code. It also required perfect 1:3 ratio code, which is rarely sent.

3.5 **Threshold Setting**

Figure 4 shows a typical histogram and 5 shows an idealized version. If point 7 can be reliably found, then the critical dot/dash and element/character
Figure 5. Threshold Histograms.
decisions can be properly made.

Blair (3) used histogram data, by assuming an initial threshold value, then calculating an empirical "goodness of separation" based on various statistical parameters. The threshold estimate was moved until the "goodness of separation" was maximized. This method requires a lot of computation and would not be suitable for real time application, but the basic idea is excellent.

The stylized histogram in Figure 5 provides the basis for the method chosen. It was found that code from different sources always had similarly shaped histograms. The most important similarities are that the dot peak is consistently higher than the dash peak, and the symbol space peak higher than the character space peak. The work space peak is not well-defined and often cannot be found. The "noise level" between peaks is lower than the peaks, and consists of elements which were poorly timed. The peaks are shown with flat tops, but this is not essential.

To find the threshold (dot/dash or symbol/character):

1. Find point 1:

   An entire histogram is searched for the largest value, point 1. If two points have this value (1 and 1'), then the rightmost one is taken because it should be closer to the threshold. This largest peak is always the dot or symbol space peak. The position and value of point 1 are noted.
2. Find point 2:

The histogram is searched from point 1 to the right for the data point which is 1/2 or less than the value at point 1. If this point does not occur before the dash peak the code is of very bad quality and cannot be decoded. Only its position is noted, as its value was used to find it.

3. Find point 3:

The first estimate of the threshold is found by calculating point 3, which is the position as far to the right of point 2 as point 2 is to the right of point 1. Its value is ignored, because it contains the bad data that this method was designed to avoid.

4. Find point 4:

The histogram is now searched for the largest data value from point 3 to the end. This will be the dash or character peak. If two bins contain this value the leftmost one is chosen as this will be closer to the threshold. The position and value are noted.

5. Find point 5:

The histogram is searched from point 4 to the left for the first bin whose value is 1/2 or less than that of point 4. If the code is very bad the noise floor could be higher than this 1/2 value, which would extend the search to the left side of the first peak. To avoid this, the search will stop at point 3, even if the desired value was not found. Code
that bad would be unreadable anyway, but a good threshold value must still be chosen to aid the slow adaptation. Only the position of this point is used.

6. Find point 6:

The second estimate of the threshold is found by calculating point 6, which is as far to the left of point 5 as point 5 is to the left of point 4. Just the position is taken as the value contains questionable data.

7. Find point 7:

Point 7 is the arithmetic average of points 3 and 6. This is the estimate of the threshold. If either line 1-2-3 or 4-5-6 has a shallow slope it is possible for point 3 to be to the right of point 6. This does not affect the choice of point 7, as it just means that 1-2-3 crosses 4-5-6 above rather than below the horizontal axis.

8.8. Space Correction:

The above seven operations are performed separately for the mark and space data, so that the only tie between the thresholds should be the overall program speed. This allows for different distributions of the marks and spaces. It also is useful for very slow code (less than 1 w.p.m.) so that long spaces can be used without long marks.

It was found however that the space data is never as well-grouped as the mark data. This becomes a problem when small code samples are obtained, as it may take a long time
to build up the space peaks properly. For this reason the space threshold estimate obtained from steps 1 to 7 is averaged with the value for the mark threshold to obtain the final space value. It was found that this helped decoding in almost all cases.

9. Character/Word Threshold:

The space between words in regular Morse code is 7 dot times long. These word spaces should appear as a third peak in the space histogram, but rarely do, so a continuation of the above methods cannot be used to find it.

The character/word threshold is set at 3.5 times the symbol/character threshold, or 7 dot times (assuming a symbol/character value of 2 dot times). Normal code would give a value between 3 and 7, but it was found that for this application the character spaces were as long as ordinary word spaces. This operation is not critical for conversational use.

The threshold setting method has been shown to find a good threshold value. No math other than integer addition, subtraction, and division by shifting is used, giving a high speed. It can be seen that a simple curve fit is actually done to find the horizontal position of the intersection of lines 1-2-3 and 4-5-6, which should be at the low point between the two distributions. Only very basic assumptions are made about the shape of the histograms,
without fixing the time ratios at 1:3 or the valley at a certain distance between peaks. Any time bins which do not have counts to a value of at least 50% of the closest peak are automatically excluded from the calculations, as are all bins on the far sides of the peaks which also do not contain information about the threshold position.

The most important objective has been achieved, and that is any elements which were incorrectly classified when first received are not averaged into the wrong place to perpetuate the initial error. Instead they are put into the histogram where they belong in time-relation to previously received elements. If they were part of a legitimate speed change, new peaks will form in the correct place to give a correct threshold, and if they were errors of low occurrence they will be completely ignored.

If reasonable quality code is received, a useable threshold is usually found after the receipt of less than a dozen marks, giving the required fast initialization time.

3.6 Out of Range Correction

If the program is running far too quickly, then the dashes will time out to the maximum of 64, rather than the ideal 24. If a mark of length 64 is detected, the timing constant is doubled (to slow the program and halve the counts) and the program is restarted with clear histograms.
This allows one to start without having to make an initial guess at the code speed. The program will restart on each mark until the timing constant is within range. This mechanism can be fooled by holding down a key to produce a long mark, so an automatic keyer should be used which cannot produce marks longer than those required. Spaces are not checked in this manner as a long space could be just a pause in sending, not a change of speed.

If the program is far too slow, a mark may only be sampled once. There is no explicit mechanism to deal with this case, as a typical key bounce would also be sampled once and a speed halving mechanism (as above) would cause the program to lock onto that noise. The normal threshold setting process can deal with this condition, for if all the marks go into the 1 bin rather than being spread around 8 and 24, the histogram fills up and empties quickly. This lets the slow adaptation and threshold mechanisms function effectively by quickly removing data stored at an earlier speed. This is a slower correction than for the opposite case above, but it is stable unless the number of noise marks greatly exceeds the number of true marks. Code of such bad quality is not expected in this application.

Flowcharts are given in Appendix B and the assembled code is in Appendix D.
CHAPTER IV

HARDWARE - AIMS AND DEVELOPMENT

4.0 Hardware Requirements

The hardware falls into three main categories:

1. Patient Interface - physical movement to audible code
2. Digital Interface - audio signal to digital input
3. Processor and Display - digital input to visible letters

The ideal system would consist of a single switch connected directly to an input port on a one chip microprocessor. An output port would drive a speaker with clean code for audio feedback, while another drove a display with the reconstructed characters. Very little other hardware would be required, as all the timing and switch debouncing would be performed in software. The computer and the display would be lightweight and low powered. The system used for development was more complex, but at all stages decisions were made with the above criteria in mind.

4.1 Patient Interface

In any manual Morse system a key is required to translate the operator's movements into a string of dots and dashes, providing an electrical signal for transmission and
an audible one for the operator. The audio feedback may come from a directly connected buzzer, monitoring of the transmitted signal, or the mechanical action of the key. Keys vary from a simple switch to dual switches with mechanisms for automatically timing perfect dots, dashes and spaces.

Normally code is sent with the hand, but in this case the subject only had reliable control over his right foot, so this point was chosen for the interface. During the largest muscle spasms it was unuseable, but good action was available at most times.

At an early point in the project, it was decided not to use a direct electrical connection between the subject and the processor. In its place the audio signal from the keyer was picked up by a microphone and conditioned for use by the computer. This approach allowed all test sessions to be recorded on a cassette recorder in any location suitable to the subject. Whether the code was "live", taped in the subject's home, or from a shortwave radio, it could all be processed the same way and stored easily for reevaluation. It also solved the problem of electrically isolating the subject from the equipment, which was line-powered except for the keyer.

One final benefit was that verbal comments on the progress of the session were recorded at the appropriate times.
and were thus more valuable than written notes. This was especially useful in analyzing the earlier sessions, as the character sent could rarely be recognized but was often framed by comments which described the true intent and the mistake.

The first key to be tried was a "Rancho"\textsuperscript{1} footswitch switch connected directly to a 12-volt battery and a Mallory Sonalert (Figure 6). The Sonalert was chosen because of its low power consumption, high efficiency and sine wave audio output. The last quality was found to be useful in triggering the phase locked loop detector described below. Ordinary buzzer sources produced complex audio waveforms which were hard to detect against background noise.

The switch could be placed in a loose-fitting shoe or sandal and was cosmetically the most pleasing, but it had a large amount of contact bounce. This was not a problem in gait studies due to the higher pressure available for contact closure, but the effect was intolerable for machine read Morse code.

It became apparent that the subject was making a valiant effort to send correct code, but he could not time out the dots and dashes properly with the single switch. It was decided that a dual action switch was needed, with

\textsuperscript{1}Developed at Rancho Los Amigos for gait studies
Figure 6. Rancho Footswitch

Figure 7. Metal Footswitch.
one position for dots and another for dashes. The author designed and built the footswitch in Figure 7. The metal loop enclosed the big toe of the right foot and pivoted in a similar way. Two microswitches sensed motion up or down from the rest position to trigger a dash or dot respectively. This orientation was chosen as the subject could press down more easily and more dots are used than dashes. The restoring force from each direction could be varied independently by changing the microswitches.

To complement this, a keyer was designed by Tony Wallace with two separate inputs which would result in an audio tone of dot or dash length when activated.

Further trials were conducted with this equipment and new problems appeared. The footswitch required the foot to remain motionless while the toe pivoted up and down. The subject however found it easier to curl his toes down and push his foot up and down, which did not activate the switch properly. The switch also had to be flexible like the Rancho switch or else much stronger, as it was easily damaged.

The keyer had no automatic space delays, so although perfect length marks were being sent, a dot and dash could be sent simultaneously. Holding a switch down produced one mark, but another often appeared on release due to contact bounce. The square wave oscillator and small speaker pro-

---

2Electronic Technologist, Chedoke Rehabilitation Centre
duced an excellent signal for the ear, but it was not
recognized over background noise by the equipment. The
length of the tones, and hence the overall speed, could not
be easily varied.

Another footswitch was designed and constructed.
(Figure 8) Two microswitches were mounted into the wooden
frame and activated by pressing the clear plastic cover plates.
This design was more rugged and easier to activate.

4.2 Automatic Keyer

The author designed and built a fully automatic keyer
which can send perfect dots or dashes, but with the addi-
tion of a perfect symbol space between them. If the foot-
switch is held down the keyer will produce a string of dots
and spaces. If it is then suddenly pushed up, the dot in pro-
gress is finished, the symbol space timed out, and the
switches sampled. If the dash switch is still activated, a
dash is begun and the switch can then be released. The
dashes are also auto-repeating if required.

There is a semi-automatic mode in which a switch has
to be released and reactivated before a mark is repeated,
although the minimum space timing is still produced.

In the manual mode either switch activates the tone
for as long as it is closed with no automatic timing.
Figure 9. Wooden Footswitch.
Since the design is fully synchronous without using monostables, the speed can be varied over a wide range with all timings (dot, dash, space) in the correct ratio. The Sonalert was used again for the advantages outlined above. The unit was built with CMOS technology and has a quiescent power drain of microwatts. For this reason and to simplify operation no power switch was included. The battery can be from 4 to 9 volts with no change in operation.

4.3 Detailed Keyer Operation

In the manual mode either switch can directly activate the output through IC1a, D3 and Q1. The trigger signal to the timing circuits is shorted to ground through D1 and SW1b.

In the automatic mode IC6 is the timing oscillator which sets the code rate. IC5 is a decade counter with 10 active high decoded outputs, clocked by IC6. If no switches have been closed:

1. IC5 is in the "0" state, closing the trigger sampler formed by IC1b and IC1c.
2. IC3a is off turning off IC6, the clock.
3. IC3b is off, turning off the output.

When a switch is closed, the trigger signal gated through IC1b and IC1c:

1. Sets IC3a to turn on IC6 and start the clock.
Figure 9. Keyer Schematic

There is no power switch.
2. Sets IC3b to turn on the output through D2 and Q1.

3. Clocks IC4a to latch the input - 1 for dash, 0 for dot.

Because the input is latched, a change in the switches will not change the timing.

IC2a, b and c select the count from IC5 which will stop the output, based on the switch latched in IC4a. A dot is stopped by the "1" count, a dash is stopped by the "3" count, giving the correct timing.

Using a dot as an example:

1. The output is on during the "0" count after triggering. This is one clock cycle long.

2. The rising edge of the "1" count, gated by IC2a, b and c, clocks IC3b to the zero state, turning off the output and ending the dot.

3. During the "1" count, the output is off and the input is locked out. This gives the required symbol space.

4. One clock cycle later, the falling edge of the "1" count clocks IC3a to the zero state, which turns off the clock and resets IC5 to the "0" state.

5. As IC5 is reset to "0" the switch sampler of IC1a and b opens again to check the switches, ending the cycle.

The dash sequence is identical, except the "3" count is used instead of the "1" count.

In the semi-automatic mode, the signal which ends the mark also clocks IC4b to the "1" state. The trigger signal
is then grounded out through the $\bar{Q}$ output and D1. When both keys have been released, IC4b is reset through ICld. The effect of this is that a switch must have been released after the completion of the mark (i.e. during the symbol space or later) and then reclosed to send another mark.

4.4 Footswitch/Keyer Results

Sequences such as "dot dash dot" (R) became "down up down" with little regard for timing. The only requirement was that a switch had to be held until the mark began, and the switch for the next mark had to be activated (or the last one released) by the time the previous symbol space was finished. Due to the very slow speeds available and the fact that the switches were locked out when not being sampled, good "noise immunity" from involuntary movements was achieved.

This equipment was tried and the only improvements needed were a reorientation and height adjustment of the upper switch, but its overall performance was finally good enough to test the software.

A final footswitch was built incorporating the needed adjustments (Figure 10) and it performed satisfactorily.

4.5 Audio Input Board

The basis of this circuit, shown in Figure 11, is IC3, a phase locked loop tone detector. This approach was chosen
Figure 10. Final Footswitch (on wheelchair mounting).
Figure 11. Audio Input Schematic
over simple amplitude detection as it gives excellent rejection of background noise. As outlined in reference 13, there are design tradeoffs involving detection bandwidth, speed of response, and noise immunity.

Switch SW1 selects a high gain microphone input or a direct input for tape playback. Gain control R5b and D3, D6 form a nonlinear input network to aid noise rejection. At low gain levels (large inputs) the signal must overcome the forward voltage drop on the diodes, so low level noise is partially inhibited. D6 was also intended to function as a level indicator, but the signal required to activate it was too large.

R22 and D4-5 clip any input to 1 volt peak to peak. If larger signals are allowed IC3 tends to detect harmonics of low frequency signals. C15 and R23-24 determine the centre detection frequency, allowing tuning from 700 Hz to 3000 Hz. C14 affects detection bandwidth and loop response time, while C16 is part of a filter on the output switch. These two capacitors were varied considerably from design values to optimize switching times and dropout rejection. R25-26 add some hysteresis to the output, and C17 increases this feedback during switching transitions.

IC1-2 are an AGC (automatic gain control) amplifier for microphone inputs. IC1a provides a gain of 100, with 3db rolloffs at 300 Hz (R1, C2) and 16 KHz (R2, C3). Q1 and
R3 are the variable gain attenuator necessary for AGC action. IClb is the amplifier inside the AGC loop, and it also lowers the upper 3db point to 10 KHz. The external gain control varies the gain of this section from 27 to 150, exclusive of the attenuator. It may seem odd to have this control inside the loop, but it limits the maximum gain for low level signals when low gain is selected. Combined with the nonlinear network of R5b, D3, D6, at low gain levels low level signals are not well amplified by IClb and are further rejected at the nonlinear network. By reducing these unwanted signals IC3 performs more reliably. High gain is available when required, but it should only be used in quiet surroundings.

D1-2 and C9-10 form a voltage doubler for AGC detection. Attack time is approximately 20ms (R11+R13, C10) and decay time 15s (R14, C10). IClc adds a dc gain of 2 to the AGC control signal.

IC2 gates the AGC signal to C10 so that it is only increased to lower the gain when a tone has been detected by IC3. This stops high room noise from lowering the gain of the system and relates it more closely to the level of the desired signal.

Tests were done in a noisy room (fan noise) using the Sonalert Keyer and a microphone pickup several feet away. As Figure 12 shows, good digital signals were produced for
Figure 12. Phase Locked Loop Tests.

a) High ambient noise (short space, long mark)

b) Low ambient noise (long space, long mark)
both low and high noise inputs.

For tracking of noisier or faster code, one could add a bandpass filter in front of the microphone amplifier with its tuning coupled to the phase locked loop. This would greatly increase noise rejection and allow other time constants to be relaxed for better tracking of fast code.

4.6 Digital Input Board

This is a simple circuit (Figure 13) to condition the detected tone from the audio board and allow the direct use of single or double keys.

The three input circuits are identical and require a switch closure to ground for a mark. R25 is a pullup resistor and R26, C25 form a low pass filter with a cutoff near 70 Hz to reduce noise from IC3 or external keys. R27-28 and IC4 form a Schmitt trigger to square the signal for digital use. D10-11 give a visual indication of external switch activation.

IC5-6 combine the three possible inputs into two data lines. Pin 1 is high for a tone mark or the dot switch and Pin 2 is high for the dash switch. Both pins high indicates an error. SW2 informs the software whether a single or double switch is being used. Pins 1, 2 and 4 connect to data lines 0, 1 and 2 respectively on an input port on the microprocessor. The dual key and SW2 sense software was not
Figure 13. Digital Input Schematic
needed or written during development, but its addition would be trivial.

4.7 Microprocessor System

The microcomputer used for development is shown in Figure 14. It is based on the single board demonstration unit marketed by Tektron Inc. using the RCA 1802 CMOS microprocessor. It was chosen as its power requirements are a few milliamps at 5 volts, and it is thus suited for the ultimate goal of a portable device. It is an 8-bit machine with sixteen general purpose 8-bit registers and a suitable instruction set. Apart from RAM and ROM it is nearly a one chip processor, as the clock oscillator, DMA, interrupt and I/O ports are provided on the chip. Further details can be found in reference 16.

The system used for development was more complex than the capabilities of the chip might imply. The unit had to drive TTL circuits, which required buffering the data and address lines. The final system would be completely CMOS and the microprocessor could drive everything directly. Two serial I/O ports, including a cassette interface, were included, as well as a hexadecimal led display and an ASCII keyboard. On the demonstration board itself was an 8-bit latched output port driving eight leds and a hexadecimal keypad which was enlarged and brought out to the front
Figure 14. Development System.
panel. Memory consisted of 2K of RAM, but the first 512 bytes were replaced with EPROM containing the monitor.

The monitor program given in Appendix C allowed editing and execution of programs, and also contained utility routines for driving the various interfaces. Operation is based on the ASCII keyboard and the hexadecimal display, but it can be run from a terminal with less flexibility.

Circuitry for this system is not given as it does not relate directly to the final system. As pointed out above, that would consist of a one board computer without the extra buffering and peripheral capabilities.
CHAPTER V

RESULTS AND CONCLUSIONS

5.0 Testing and Results

Testing involved two main areas, testing with the patient and without. Both the hardware and the software were initially developed using the general rules for Morse code, and then modified as required.

The footswitch progressed from a single switch and a buzzer to a dual switch designed for the feet, plus an automatic keyer. The final combination of an adjustable wooden frame and microswitches covered in plexiglass proved to be functionally adequate and quite rugged. Required pressure for activation was easily varied by inserting foam rubber under the plexiglass. For a patient with weaker movements and more control it should be possible to place a single or dual switch entirely within a shoe using the Rancho type switch.

The automatic keyer was found to be necessary due to the subject's relatively poor control compared to a normal Morse operator. In this respect Morse code with its timing requirements might not be the ideal code for the cerebral palsied. Again, for a subject with better control the keyer
may not be needed, or could be incorporated into the software.

The program began as a short decoder of ideal code with fixed dot/dash ratios. This performed perfectly in tests using perfect code from another 1802 system, but failed miserably using hand sent code. The final program was much longer than anticipated (as usual!), but produced readable code. Some tests were conducted using code from a shortwave radio using the phase locked loop detector described in the hardware section. This code was being sent at rates up to 20 times those anticipated for this project, but readable code was still produced. One caution to others trying this test is that amateur radio operators often use a bewildering array of abbreviations which at first appear to be bad decoding. Due to international regulations, it is unfortunately illegal to reproduce here any of those tests.

Tests with the patient were quite brief due to logistic problems. These sessions were taped however and used repeatedly. It was found that for use as a conversational aid extra non-standard characters had to be added to do things such as blank the display and perform a carriage return. Due to the very long times between letters the automatic letter space should be deleted and replaced with a special space code, although this was not done.
5.1 Trial Unit

Near the end of the project a summer student built a portable decoder based on the work presented here. (15) The result, shown in Figure 15, consisted of the footswitch and keyer, the processor, the battery and charger, and a 20 character alphanumeric display. The size of the battery was due mainly to the fluorescent display.

Several trials were conducted with this apparatus on a wheelchair and it fulfilled the requirements. At the present time, intermittent electrical failures are preventing its use, but a more reliable unit will be produced.

5.2 Conclusions and Recommendations

The main objectives of the project were met, in that Morse code was shown to be a feasible means of portable communication for a cerebral palsied person.

Improvements can now be made all round, due to a better understanding of the problem, and more compact and lower power technology. Studies of the histograms generated by the user should be made so that the decoding algorithm can be tailored to the user. Convenience features can be included, such as the facility for the unit to operate as a computer terminal. A liquid crystal display would allow week long operation on a single charge, or a smaller battery. The actual switch required depends on the individual user, as with
Figure 15. Trial Unit.
all aids, but a lighter and more visually pleasing switch could be designed.

It should be noted that most of the above improvements are aesthetic (editing, lower weight, etc.) rather than functionally essential. It is the author's experience however that unless an aid is almost invisible in its active and passive states it is apt to be rejected by the user.

In closing, an aid is only useful if it is used! It is the author's hope that further work will be done in this area, both in clinical trials and final placing of devices with the handicapped.
APPENDIX A

MORSE CODE CHARACTERS

AND

PROGRAM COMMANDS
### MORSE CODE CHARACTERS

<table>
<thead>
<tr>
<th>Letter</th>
<th>Morse Code</th>
<th>Number</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>. -</td>
<td>1</td>
<td>Period</td>
</tr>
<tr>
<td>B</td>
<td>- . .</td>
<td>2</td>
<td>Comma</td>
</tr>
<tr>
<td>C</td>
<td>- - .</td>
<td>3</td>
<td>Question</td>
</tr>
<tr>
<td>D</td>
<td>- . .</td>
<td>4</td>
<td>Error</td>
</tr>
<tr>
<td>E</td>
<td>. . .</td>
<td>5</td>
<td>Colon</td>
</tr>
<tr>
<td>F</td>
<td>S . .</td>
<td>6</td>
<td>Semicolon</td>
</tr>
<tr>
<td>G</td>
<td>T -</td>
<td>7</td>
<td>Bracket</td>
</tr>
<tr>
<td>H</td>
<td>U . .</td>
<td>8</td>
<td>Backslash</td>
</tr>
<tr>
<td>I</td>
<td>V . .</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>W . .</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>X - . .</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>Y - . .</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Z - . .</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### PROGRAM COMMANDS FROM ASCII KEYBOARD

/ Restart at 03E0, call monitor for new speed
H Output histogram immediately, wait for more commands
K Send "COPY" command to graphics terminal, wait
M Restart at 03F6, some histogram, speed and thresholds
N Set non-auto mode, restart at 03F6 (cancel a "Z")
X Clear histograms and continue
Y Divide histogram data by 2 and continue
Z Set automatic mode (output a histogram on any overflow)
PROGRAM OVERVIEW

START

reset all counters and histograms

input initial value for time constant

TIMER entry

delay for 1 time constant

is key up Yes jump to KEYUP routine

No (enter key down routine)

was key down before Yes

No

put keyup time into histogram

is timer full

Yes

No

Increment key down timer
PROGRAM OVERVIEW (continued)

enter KEYUP routine

was key up before

Yes

No

was previous mark

Dash

Dot

put "0" into character register

put "1" into character register

? perfect dot

Yes

No

adjust time constant slightly to correct overall speed

put key down time into histogram

increment count of marks in present character

(continued next page)
PROGRAM OVERVIEW (continued)

check keyup timer

inter character space

Yes

No

decode Morse and output character

Increment keyup timer

calculate new decision threshold values

is timer full

Yes

No

increment keyup timer

any keyboard data

No

Yes

take appropriate action

jump to Timer entry continue timing
MORSE 03DE-03FF

**enter RSTRT 03DE**
(full restart)

00 in RC.0 monitor flag for data entry

**enter MORSE 03EO**
(starting point)

03E6 in R3, 3 in X set up R3 as program counter

output "FO" to format serial port 2

RESET Q flag

CALL SIN this calls the monitor as a subroutine to input initial speed to RD, type "R" to return here

**enter RSTR2 03EF**
(partial restart)

10 in RE.1, RE.0 = mark & space threshold preset

CALL MASG with Q = 0, set histograms to zero

**enter RSTR1 03F6**
(semi restart)

CALL CRLF output "CR" & "LF" to show (re)start

(continues on next page)

**reset some registers to 00**
RC.0 = time of keyup
R9.0 = time of key down
RC.1 = no. of marks in a character
R9.1 = dot/dash (0/1) shift register

**enter TIME 0400**
(timing loop)

RD in RF get timing parameter, prop. to 1/speed

Is RF = 0000 N decrement RF

Y

input parallel port 1, lsb = 0, key up; lsb = 1, down

Is keyup Y JUMP to UP

N

Is keyup = 00 Y JUMP to STDN key still down

N (key just down)

Is keyup = 40 Y (overflow)

N (time is O.K.)

CALL UBIN bin keyup time

Is Q set Y JUMP to AUT bin is full

N
The "AUT" section of code is in a difficult position. It acts like a subroutine, but in fact "Q" is used to determine the "return" address.

The "UP" section of code:

- **07EE in R7 = auto-flag memory address**
  - Get M(R7) = auto-flag
  - 07F0 in R7 = I/O add
  - If flag "Z", JUMP to HIST output hist
  - If Q set, RESET Q
  - JUMP to UP2

- JUMP to STUP

- **R9.0 - RE.1 in D = key down - mark threshold**
  - If D = 00, Y (mark = dash)
  - N (mark = dot)
  - Shift a 0 into lsb of R9.1 signifying last mark = dot
  - R9.0 - 08 in D = mark - perfect dot time
  - Shift a 1 into lsb of R9.1 cause last mark was dash
  - R9.0 - 18 in D = mark - perfect dash time
  - If D = 00, Y (speed is OK)
  - N (next page)
  - RD + 1 in RD this raises timing constant = slower
  - RD - 2 in RD effect'ly -1
CALL DBIN bin down time

is Q set

Reset Q

JUMP to AUT bin is full

increment RC.1 = no. of marks in this character

enter UP2 0456 return from AUT, no histogram

00 in R9.0 reset key down timer

is keyup = 40

Y (timer full)

is RC.1 = 00

Y (no marks, no letter)

N

is keyup = RE.0

Y

JUMP to LOUT output letter

N

find 3.5 * RE.0 = inter-character space, ICS

is keyup = ICS

output "20" = ASCII "space"

N

increment RC.0 = keyup timer

any keyboard data

Y

JUMP to KCHK

N

JUMP to TIME continue normal timing
CALL DBIN bin down time

is Q set

Y Reset Q JUMP to AUT bin is full

N enter UP2 0456 return from from AUT, no histogram

increment RC.1 = no. of marks in this character

00 in R9.0 reset key down timer

is keyup = 40 Y (timer full)

N

is RC.1 = 00 Y (no marks, no letter).

N

is keyup = RE.0 Y JUMP to LOUT output letter

N

find 3.5 * RE.0 = inter-word space, IWS

is keyup = IWS Y output "20" = ASCII "space"

N

increment RC.0 = keyup timer

any keyboard data Y JUMP to KCHK

N JUMP to TIME continue normal timing
output R9.1 to POP1 put morse register on leds for visual check (optional)

is RC.1 =06

N

Y (character has 6 marks)

is letter "?"

load "?"

N

is letter ":"

load ":"

Y

N (not "?" or ":")

N (6 or fewer)

is RC.1 >06

Y

overrun

R9.1 AND 3F in RF.0 mask in lower 6 morse bits, 1 of 64 addresses

05 in RF.1 = high address

load M(RF) = desired ASCII letter

output character

Increment RC.0 = keyup timer

RC.0 in M(07EF) save keyup timer

JUMP to THRMA find new thresholds
ASCII data into D, M(RX)

- If data is "/", JUMP to RSTRT full restart
- If data is "H", JUMP to HIST for histogram
- If data is "K", Y
  - 0210 in RA = address for serial port 2 routine
  - Output copy command "ESC", "ETB" to graphics terminal
- If data is "M", Y
  - JUMP to RSTRL semi restart
- N

RESET Q for hist = 00

- If data is "Y", SET Q for hist/2
- If data is "X", CALL MASG fix hist
- N
- JUMP to RSTRL

If 07EE in R7 = auto flag memory address

- "Z" = auto, "N" = n.auto
- 07F0 in R7 = I/O address
- JUMP to TIME continue decoding after "N", "Z"
MASG 03C0-03D4

enter

0580 in RF = start add of raw histogram data

is RF.0 = 00

Y RETURN (RF=0600)

N

is Q set

Y get M(RF) = hist data

N 00 in D

data/2 in D

D in M(RF) = fixed data

increment RF

UBIN, DBIN 0540-055E

enter UBIN 0540

bin key up timer

80 + RC.0 in D = offset + key up time = low add

enter DBIN 0545

bin key down timer

is timer > 40

Y

JUMP to BIGM

N

CO + R9.0 in D = offset + key down = low add

D in RF.0 = low address = actual time + offset

05 in RF.1 = high address

M(RF) in D = present count

is count = 3F

Y SET Q bin full

N

increment count, store it back into M(RF)

JUMP to RSTR2 semi restart

JUMP to SMTH this smooths histogram data
SMTH 0760-079F

enter

0580 in RF = start add of raw histogram data

0680 in RO = start add of smoothed data

0 in X RO used for math

get data(N) from M(RF) (N is a dummy index)

multiply data(N) by 2

D in M(RO) = 2•data(N) into smooth data(N)

decrement RF = data(N-1)

get data(N) from M(RF)

M(RO) + 2 in D add 2 to correct round-off

D/4 in M(RO) divide and store final result

increment RO = next smooth bin

7 in X (normal value)

CALL MASN this divides histograms by 2 as Q=1

output "CR","LF" & "">

return

Note: smoothed data(N) = (data(N-1) + 2•data(N) + data(N+1) + 2)/4
**LTOR 0710-0723**

**entry:** half peak in RC.1

**start address of search in R9**

00 in RF.0

**exit:** RF.0 = # of bins from peak to first bin <= 1/2 peak from left to right

**RTOL 0728-073F**

**entry:** half peak in RC.1

**start address in R9.0**

00 in RF.0

lowermost bin # in RE.0 for spaces, RE.1 for marks

**Q = 0**, spaces; **Q = 1**, marks

**exit:** RF.0 = # of bins from peak to first bin <= peak from right to left

---

**Note:** LTOR and RTOL are very similar and differ in direction of scan and overrange criteria.
conditions on entry:
Q = 0 to find right maximum
Q = 1 to find left maximum
RF.0 has lowest bin number to be searched.
R9 has that memory address.

conditions on exit
RC.0 has the bin number with the maximum data value.
RC.1 has that data value.

<table>
<thead>
<tr>
<th>Enter</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 in RC.1, RC.0 reset max. value and bin #</td>
</tr>
<tr>
<td>is Q set N 00 in RF.0 right max Y</td>
</tr>
<tr>
<td>9 in X R9 is used as a memory pointer for math</td>
</tr>
</tbody>
</table>

| RC.1 - M(R9) in D find current max. - new data |
| is current new Y (never change if current > new) N |
| is Q set Y (max, change if < only) |
| (r. max, N change on < or = ) |
| is current = new Y (left max, no change on = ) N (must be <, change) |

| M(R9) in RC.1 hold new data as max. value |
| RF.0 in RC.0 hold new bin # as max. bin # |
| Increment RF point (logically) to next bin |
| Increment R9 point (memory) to next bin |

| is R9.0 = 00, BF Y (R9 = 0600 or 05BF) N (continue) 7 in X as usual |
| RETURN done |
ADD50  0700-070F

entry: peak value in RC.1
peak bin # in RC.0
Q = 1 for marks
Q = 0 for spaces

exit: half peak in RC.1
actual memory add.
of peak in R9.0

```
enter

00 in RF.0  reset
bin counter

(RC.1)/2 in RC.1  find
half maximum value

is
Q
set

Y (marks)
N (spaces)

load 80 =
space offset

load CO =
mark offset

D + RC.0 in R9.0  offset
+ peak bin # = low address

RETURN
```
06CO in R9 start address of smoothed mark data

reset Q to find right maximum in FMAX

CALL FMAX find right maximum (dot peak, point 1)

set Q for mark routine in ADD50

CALL ADD50 fix data from FMAX

SUB LTOR find bin 50% or less than dot peak (point 2)

RC.0 + 2*RF.0 in D, RF.0 (counter) & RE.1 (left stop) find bin twice as far from peak as 50% bin, estimate of threshold based on dot peak and start point for dash peak search (point 3)

D + CO in R9.0 convert this to a memory address

set Q to find left maximum in FMAX

CALL FMAX find left maximum (dash peak, point 4)

set Q for mark routine in ADD50

CALL ADD50 fix data from FMAX

SUB RTOL find bin 50% or less than dash peak (point 5)

RC.0 - 2*RF.0 in D find bin twice as far from dash peak as 50% bin, estimate of threshold based on dash peak, (point 6)

(continued next page)
THRM (continued previous page)

\[(\text{RE.1} + D)/2\] in \text{RE.1} average two estimates (dot, dash) for mark threshold, store it in \text{RE.1} (point 7)

JUMP to THRS

THRS 0640-067F

This routine is very similar to THRM, and differs in detail only. IRCS refers to the intercharacter space, and IACS to the intracharacter space.

enter

0680 in R9 start address of smoothed space data
reset Q to find right maximum in FMAX
CALL FMAX find right maximum (IACS peak, point 1)
reset Q for space routine in ADD50
CALL ADD50 fix data from FMAX
SUB LTOR find bin 50\% or less than IACS peak (point 2)
RC.0 + 2*RF.0 in D, RF.0 (counter) & RE.0 (left stop) find bin twice as far from peak as 50\% bin, estimate of threshold based on IACS peak and start point for IRCS peak search (point 3)
D + 80 in R9.0 convert this to a memory address
set Q to find left maximum in FMAX
CALL FMAX find left maximum (IRCS peak, point 4)
THRS (continued previous page)

reset Q for space routine in ADD50

CALL ADD50 fix data from FMAX

SUB RTOL find bin 50% or less than IRCS peak (point 5)

RC.0 - 2*RF.0 in D find bin twice as far from IRCS peak as 50% bin, estimate of threshold based on IRCS peak, (point 6)

(RE.0 + D)/2 in RE.0 average two estimates (IACS,IRCS) for space threshold, store it in RE.0

(RE.0 + RE.1)/2 in RE.0 average this estimate for the space threshold with estimate for the mark threshold, due to the low "quality" of the space information, store it in RE.0 (point 7)

JUMP to CLUP

CLUP :0780-07BD

SUB DISPY display new thresholds

00 in R9.0 reset R9.0 for use as key down timer

00 in RC.1 reset RC.1 to count number of marks/character

01 in R9.1 preset morse shift register with leading 1

M(07EF) in RC.0 restore key up time value to RC.0

JUMP to TIME thresholds found, continue regular timing
XYOU1 02E0-02F1

entry: 8 bit x coordinate in M(07F1), y in M(07F2)

enter

M(07F1) in R0.0
get x coordinate

M(07F2) in R0.1, D
get y coordinate

SHIFT D right 3 times
get high 5 bits of y coordinate

D OR 20 add "high y byte"
identifier

output "high y byte"

R0.1 in D
get y coordinate

SHIFT D left 2 times
set 2 lower bits to 0

D in M(07F0) save intermediate y value

RETURN

At this point the intermediate y value in M(07F0) can have the two lowest bits set by the main program. This would allow the full 10 bit accuracy of the graphics terminal to be used.

XYOU2 02F8-030D

entry: intermediate y value in M(07F0)

enter

M(07F0) in D get intermediate y value

D AND 1F mask in low 5 bits

D OR 60 add "low y byte" identifier

output "low y byte"

R0.0 in D get x coord.

SHIFT D right 3 times
get high 5 bits of x

D OR 20 add "high x byte" identifier

output "high x byte"

R0.0 in D get x coord.

SHIFT D left 2 times
set 0 into 2 lowest bits

D OR 40 add "low x byte" identifier

output "low x byte"

RETURN
entry: RC.0 = F1 for x axis, F2 for y axis; a low mem. add. 
Q = 0 to letter in +ve direction (x or y) and v.v. 
RC.1 = screen address increment per letter 
M(07F1) = x coordinate, M(07F2) = y coordinate 
R9.0 = last position to be lettered, +/- 1

enter

00 in RF.0 first letter

Q set

OF in RF.0 first letter

output GS set terminal into graphics mode

CALL XYOU1, XYOU2 output x & y coordinates

output US set alphanumeric mode at x & y coordinates

RF.0 AND OF in RF.1 put low 4 bits of axis count in RF.1

CALL 1HOUT output ASCII equivalent to letter axis

RC.0 in R7.0 R7 now points to x or y value to be changed

M(R7) + RC.1 in M(R7) increment screen address as needed

increment RF.0 +ve direction

Q set

decrement RF.0 -ve direction

RF.0 = R9.0

N (continue)

Y (finished this axis)

RETURN
CALL CRLF output a carriage return & line feed to port 1

0210 in RA point RA to routine for serial port 2

output "BEL" inform operator of histogram

output "ESC" & "FF" erase graphics screen

FFFF in RF set up delay loop while screen clears

is RF.1 =00 N

F1 in RC.0 letter x axis 04 in RC.1 x increment
04 in M(07F1) x offset 61 in M(07F2) y offset
3F in R9.0 last count + 1 0 in Q positive increment

CALL LTTR letter x axis in positive direction, 0 to 3E

F2 in RC.0 letter y axis 05 in RC.1 y increment
00 in M(07F1) x offset 11 in M(07F2) y offset
FF in R9.0 last count - 1 1 in Q negative increment

CALL LTTR letter negative y axis, F to 0

F2 in RC.0 letter y axis 05 in RC.1 y increment
00 in M(07F1) x offset 66 in M(07F2) y offset
10 in R9.0 last count + 1 0 in Q positive increment

CALL LTTR letter y axis in positive direction, 0 to F

JUMP to TEKH output data to labelled axes
entry: Q = 0, space data is output first

CALL XYOU1, XYOU2 dark vector to start of next bar (x,0)

M(RC) * 1.25 in D scale smoothed histogram data

CALL XYOU1 output "high y byte", upper 5 bits of 10

M(RC) AND 03 in D get data again, mask in 2 lowest bits

CALL XYOU2 output "low y byte", lower 5 bits of 10, "low x byte" and "high x byte" (only 8 significant bits)
cont. TEKH 0351 from prev. page

Increment RC point to next histogram data byte

M(07F1) + 04 in M(07F1) increment x coordinate

\[ \text{is } \quad RC,0 < \text{BF} \quad \Rightarrow \text{JUMP to TEK1} \]

\[ \text{spaces not done} \]

\[ \text{N (spaces done)} \]

Set Q for mark output

\[ \text{is } \quad RC,0 = \text{BF} \quad \Rightarrow \text{JUMP to TEKH back to start, begin mark output} \]

\[ \text{N} \]

\[ \text{is } \quad RC,0 = \text{FF} \quad \Rightarrow \text{JUMP to TEK1 marks not done} \]

\[ \text{Y (done)} \]

Reset Q

\[ \text{JUMP to OUPAR output parameters and headings} \]
OUPAR 0370-03AF

enter

00 in M(07F1) = X coord
BA in M(07F2) = Y coord

output 1D (control "GS")
to select graphics mode

CALL XYOUT,XYOUT send
coords as dark vector

output 1F (control "US")
to set alpha mode

02B0 in RC = start
address of headings

CALL HEAD output "SPEED"

CALL RDOUT output RD,
the speed parameter

02B7 in RC = next heading

CALL HEAD output "MARK THRESHOLD"

RE.O in RF.1 CALL 2HOUT
output mark threshold

02CB in RC = next heading

CALL HEAD output "SPACE THRESHOLD"

HEAD 03B6-03BC

enter

start address of ASCII
list is in RC

get ASCII data from
M(ASCII), increment RC

is data = 00

Y

RETURN

N

output ASCII data to
selected serial port

RE.0 in RF.1 CALL 2HOUT
output space threshold

0201 in RA select
serial port 1 again

JUMP TO KCHK this is
the end of the histogram
APPENDIX C

MONITOR PROGRAM
MONITOR OPERATION

Most of the monitor resides in EPROM from 0000 to 01FF. The two serial output drivers sit in RAM from 0200 to 021F and must be reloaded before use. To start the monitor hit "Reset" and "Run" on the front panel. The available commands are all single key commands which are immediately acted upon, no carriage return is required. All addresses or data are displayed on the 8 digit hexadecimal display above the ASCII keyboard. RD is displayed in the left four digits and RE in the right.

Key Function
break ;Interrupts the processor and restarts the monitor at 0000. This may be used at any time to regain control from a bad program which has NOT used R1.
space ;Normally numeric data goes to RD, but after the space bar is pressed data goes to RE. After the next command data flow returns to RD.
0-9, A-F ;Enters hexadecimal numbers to RD or RE.
? ;The data in M(RD) is displayed in RE.0, RE.1=00
I ;Use after "?" to display successive locations.
! ;The data in RE.0 is entered into M(RD). If "!" is hit again, RD is advanced by 1 and RE.1=00. New data can now be entered in RE.
$ ;Start execution of a program at M(RD) with R0 as RP. R3 should be made RP at the start of that program.
P ;All the data from M(RD) to M(RE) is moved down by one to M(RD+1) to M(RE+1). This is useful but any jump addresses must be corrected by hand.
G ;Any data from serial port 1 is dumped into M(RD) to M(RE). When (RE-RD) bytes have been collected or if there were any errors the program returns to the monitor. This is usually used for loading from a cassette.
W ;All the data from M(RD) to M(RE) is dumped out serial port 1. This is usually used to record a cassette.
R ;If the monitor was called as a subroutine by a running program this command is used to return to that program. This is useful for entering data during execution.
TT ;This is a two key command. The data from M(RD) to M(RE) is output to serial port 1 as double hexadecimal ASCII characters, with each pair followed by "CR" & "LF".
TP ;This is used to obtain listings on the Teletype.

Many useful routines are hidden in the monitor and can be called by other programs. The whole monitor can be called, with return effected by "R". Some routines will return automatically if "Q" is set.
PORTS & FLAG ASSIGNMENTS

The three "N" lines from the 1802 are decoded to seven lines (called "DEC N") to allow direct access to seven I/O ports.

<table>
<thead>
<tr>
<th>Code</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>61</td>
<td>OUT SER1</td>
<td>This is serial port 1, a UART which drives an FSK cassette, 20 ma loop TTY, and EIA RS 232 interfaces. Baud rate is 110 or 600, format is 7 or 8 bits with optional parity, selectable by front panel switches. EF3 = 1 signifies transmitter busy, EF3 = 0, ready.</td>
</tr>
<tr>
<td>69</td>
<td>INP SER1</td>
<td>This is serial input port 1, and it is configured the same way as the output port. The three input interfaces are OR'd, so only one should be used at a time. Input and output may proceed simultaneously. UART error flags are OR'd onto a flag: EF1 = 1 means error, EF1 = 0, no error. EF2 = 0 means new data ready, EF2 = 1, no data. There is a front panel switch to connect EF2 to the DMA so tapes may be loaded with no bootstrap at 0000.</td>
</tr>
<tr>
<td>62</td>
<td>OUT POP1</td>
<td>Parallel output port 1, a set of 8 leds on the CPU card and a dip connector.</td>
</tr>
<tr>
<td>6A</td>
<td>INP HEX</td>
<td>This is the hex keyboard on the front panel. If the CPU is in the load mode, each pair of hex digits will be DMA'd into M(R0). It can be read with 6A.</td>
</tr>
<tr>
<td>63</td>
<td>OUT DISP</td>
<td>Output to the hex led display. The data must be formatted by subroutine DISPY. This display is used extensively by all programs.</td>
</tr>
<tr>
<td>6B</td>
<td>INP ASKEY</td>
<td>Input the ASCII encoded keyboard. EF4 = 1 means new data, EF4 = 0, no data.</td>
</tr>
<tr>
<td>64</td>
<td>OUT POP2</td>
<td>This is a one bit (DO) output port used by the morse program to output the sampled key as a check on sampling rate. It is not essential.</td>
</tr>
<tr>
<td>6C</td>
<td>INP 4</td>
<td>Not used.</td>
</tr>
<tr>
<td>65</td>
<td>OUT 5</td>
<td>Not used.</td>
</tr>
<tr>
<td>6D</td>
<td>INP 5</td>
<td>Not used.</td>
</tr>
<tr>
<td>66</td>
<td>OUT SER2</td>
<td>Output data to serial port 2. This port has 20 ma TTY and EIA RS 232 interfaces. The baud can be set on the back panel from 75 to 4800.</td>
</tr>
<tr>
<td>6E</td>
<td>INP SER2</td>
<td>Input data from serial port 2.</td>
</tr>
<tr>
<td>67</td>
<td>OUT CON2</td>
<td>This sets the control register on serial port 2. D7=PI, D6=SBS, D5=WLS1, D4=WLS2, D3=PS. D2-D0 are not used. Output &quot;F0&quot; to set 8 data bits, no parity and 2 stop bits.</td>
</tr>
<tr>
<td>6F</td>
<td>INP PIP1</td>
<td>D7=1 if serial port 2 is ready to send, D5=1 for new data ready, D5=1 for any errors, D4 and D3 are not used. D2 - D0 connect to the morse interface, D0=1 for key down.</td>
</tr>
</tbody>
</table>
### LABELS

Some labels are entry points of significance, but most are merely for internal branches and deserve no comment.

<table>
<thead>
<tr>
<th>Add.</th>
<th>Label</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>START</td>
<td>Beginning of monitor, resets all required registers. The interrupt is not enabled until this is complete. Many routines and the interrupt come to this point.</td>
</tr>
<tr>
<td>0030</td>
<td>CHECK</td>
<td>If Q=1, executes an SCRT RETURN from here.</td>
</tr>
<tr>
<td>0033</td>
<td>SIN</td>
<td>Enter monitor here as a subroutine.</td>
</tr>
<tr>
<td>0036</td>
<td>DIN</td>
<td>Enter here if input already in RF.1.</td>
</tr>
<tr>
<td>003E</td>
<td>S1</td>
<td></td>
</tr>
<tr>
<td>004B</td>
<td>S1B</td>
<td></td>
</tr>
<tr>
<td>0070</td>
<td>ATOF</td>
<td>Jump here if monitor recognizes hex A to F</td>
</tr>
<tr>
<td>0074</td>
<td>NUM</td>
<td>and here if 0 to 9.</td>
</tr>
<tr>
<td>007D</td>
<td>MOSH</td>
<td></td>
</tr>
<tr>
<td>008D</td>
<td>LRE</td>
<td></td>
</tr>
<tr>
<td>0098</td>
<td>SHOV?</td>
<td>Here from look routine to examine next add.</td>
</tr>
<tr>
<td>00A4</td>
<td>LOGAN</td>
<td>Routine to examine memory. Not a subroutine.</td>
</tr>
<tr>
<td>00A5</td>
<td>LOOK</td>
<td></td>
</tr>
<tr>
<td>00B8</td>
<td>RENM</td>
<td>Routine to modify memory. Not a subroutine.</td>
</tr>
<tr>
<td>00C0</td>
<td>MOD</td>
<td></td>
</tr>
<tr>
<td>00DB</td>
<td>SOVER</td>
<td></td>
</tr>
<tr>
<td>00DC</td>
<td>SRX4</td>
<td>SEP subroutine to shift D right four places.</td>
</tr>
<tr>
<td>00E2</td>
<td>HOVER</td>
<td></td>
</tr>
<tr>
<td>00E3</td>
<td>HXOUT</td>
<td>SEP subroutine to output to hardware display</td>
</tr>
<tr>
<td>00F4</td>
<td>INPUT</td>
<td>SCRT subroutine waits for input from serial port 1 or the ASCII Keyboard. Data returned in M(RX) and RF.1.</td>
</tr>
<tr>
<td>00F9</td>
<td>KB?</td>
<td></td>
</tr>
<tr>
<td>00FC</td>
<td>GOTIT</td>
<td></td>
</tr>
<tr>
<td>0100</td>
<td>EXITC</td>
<td></td>
</tr>
<tr>
<td>0101</td>
<td>SUB</td>
<td>SEP subroutine for subroutine calls by SCRT.</td>
</tr>
<tr>
<td>0112</td>
<td>EXITR</td>
<td></td>
</tr>
<tr>
<td>0113</td>
<td>RETRN</td>
<td>SEP subroutine for SCRT subroutine exits.</td>
</tr>
<tr>
<td>0120</td>
<td>WRITE</td>
<td>See RCA 1802 Users Manual for SEP and SCRT. Cassette or paper tape dump. Not subroutine.</td>
</tr>
<tr>
<td>0121</td>
<td>TFRE?</td>
<td></td>
</tr>
<tr>
<td>0134</td>
<td>PUSH</td>
<td>Routine to make a hole. Not a subroutine.</td>
</tr>
<tr>
<td>0136</td>
<td>MPUSH</td>
<td></td>
</tr>
<tr>
<td>0147</td>
<td>PDONE</td>
<td></td>
</tr>
<tr>
<td>0150</td>
<td>GET</td>
<td>Routine to input tapes. Not a subroutine.</td>
</tr>
<tr>
<td>0157</td>
<td>DELAY</td>
<td></td>
</tr>
<tr>
<td>015D</td>
<td>SER?</td>
<td></td>
</tr>
<tr>
<td>0161</td>
<td>GMORE</td>
<td></td>
</tr>
<tr>
<td>016C</td>
<td>MSER?</td>
<td></td>
</tr>
<tr>
<td>0170</td>
<td>GDONE</td>
<td></td>
</tr>
</tbody>
</table>
Add. | Label | Comments |
--- | --- | --- |
0178 | TTY | ;Routines for hex/ASCII dumps. Not subroutine |
0188 | MORTY | ; |
018B | NOTTP | ; |
01A1 | NOADD | ; |
01CE | ROUT | ;SCRT subroutine outputs RD as 4 hex/ASCII |
01DA | CRLF | ;SCRT subroutine outputs an ASCII "CR" & "LF" |
01E2 | 2HOUT | ;SCRT output subroutine, 2 hex/ASCII from RF.1 |
01F2 | 1HOUT | ;SCRT output subroutine, 1 hex/ASCII from RF.1 |
0200 | S1DON | ; |
0201 | SOUT1 | ;SEP subroutine outputs D to serial port 1. |
0202 | S1OK? | ; |
020F | S2DON | ; |
0210 | SOUT2 | ;SEP subroutine outputs D to serial port 2. |
0214 | T20K? | ; |

**MONITOR REGISTER ASSIGNMENTS**

<table>
<thead>
<tr>
<th>Register</th>
<th>Initial Value</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>0000</td>
<td>;Reset internally by 1802; initial program counter (RP)/ RP after &quot;$&quot; command/ DMA pointer.</td>
</tr>
<tr>
<td>R1</td>
<td>0000</td>
<td>;RP after interrupt to restart monitor, reset before interrupt reenabled. To ensure correct interrupt action, R1 should NEVER have other uses.</td>
</tr>
<tr>
<td>R2</td>
<td>07E0</td>
<td>;Stack pointer for SCRT addresses.</td>
</tr>
<tr>
<td>R3</td>
<td>0007</td>
<td>;RP for most programs.</td>
</tr>
<tr>
<td>R4</td>
<td>0101</td>
<td>;RP for SEP routine CALL for SCRT.</td>
</tr>
<tr>
<td>R5</td>
<td>0113</td>
<td>;RP for SEP routine RETURN for SCRT.</td>
</tr>
<tr>
<td>R6</td>
<td>----</td>
<td>;Scratchpad for CALL and RETURN.</td>
</tr>
<tr>
<td>R7</td>
<td>07F0</td>
<td>;RX for most programs. 07F0 is a free location for I/O and R7 should point here when not in use. The area from 07E1 to 07FF is available as a stack for data, but only 07F0 is used by the monitor.</td>
</tr>
<tr>
<td>R8</td>
<td>00DC</td>
<td>;RP for SEP routine SRX4.</td>
</tr>
<tr>
<td>R9</td>
<td>----</td>
<td>;Scratchpad</td>
</tr>
<tr>
<td>RA</td>
<td>0201</td>
<td>;RP for SEP routine SOUT1.</td>
</tr>
<tr>
<td>RB</td>
<td>00E3</td>
<td>;RP for SEP routine HXOUT.</td>
</tr>
<tr>
<td>RC</td>
<td>--00</td>
<td>;Scratchpad, RC.0 is used as a flag.</td>
</tr>
<tr>
<td>RD</td>
<td>----</td>
<td>; Basically scratchpad, but since RD is output to the led display, it is used by many routines for addresses or data. The display should be updated each time RD is modified.</td>
</tr>
<tr>
<td>RE</td>
<td>----</td>
<td>;Same as RD.</td>
</tr>
<tr>
<td>RF</td>
<td>----</td>
<td>;Scratchpad.</td>
</tr>
<tr>
<td>Add. Code</td>
<td>Label</td>
<td>Mnemonic</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>0000 F8 00</td>
<td>START</td>
<td>LDI 00</td>
</tr>
<tr>
<td>0002 B3</td>
<td>PHI 3</td>
<td>LDI 07</td>
</tr>
<tr>
<td>0003 F8 07</td>
<td>PHI 3</td>
<td>LDI 07</td>
</tr>
<tr>
<td>0005 A3</td>
<td>PLO 3</td>
<td></td>
</tr>
<tr>
<td>0006 D3</td>
<td>SEP 3</td>
<td></td>
</tr>
<tr>
<td>0007 7A</td>
<td>REQ</td>
<td>If monitor restarted from</td>
</tr>
<tr>
<td>0008 F8 00</td>
<td>PHI 1</td>
<td>Initialize the registers</td>
</tr>
<tr>
<td>001A A1</td>
<td>PHI 8</td>
<td></td>
</tr>
<tr>
<td>000D AC</td>
<td>PHI B</td>
<td>0000 R1</td>
</tr>
<tr>
<td>000E BB</td>
<td>PHI I</td>
<td>07E0 R2</td>
</tr>
<tr>
<td>000F F8 01</td>
<td>PHI 4</td>
<td>0013 R5</td>
</tr>
<tr>
<td>0011 A4</td>
<td>PHI 5</td>
<td>07F0 R7</td>
</tr>
<tr>
<td>0012 B4</td>
<td>PHI A</td>
<td>00DC R8</td>
</tr>
<tr>
<td>0013 B5</td>
<td>PHI A</td>
<td>07E0 R2</td>
</tr>
<tr>
<td>0014 AA</td>
<td>PHI B</td>
<td>00E3 R8</td>
</tr>
<tr>
<td>0015 F8 02</td>
<td>PHI B</td>
<td>00E3 R8</td>
</tr>
<tr>
<td>0017 BA</td>
<td>PHI B</td>
<td>00E3 R8</td>
</tr>
<tr>
<td>0018 F8 E0</td>
<td>PHI B</td>
<td>00E3 R8</td>
</tr>
<tr>
<td>0019 A5</td>
<td>PHI B</td>
<td>00E3 R8</td>
</tr>
<tr>
<td>001A A2</td>
<td>PHI B</td>
<td>00E3 R8</td>
</tr>
<tr>
<td>001B F8 DC</td>
<td>PHI B</td>
<td>00E3 R8</td>
</tr>
<tr>
<td>001C A8</td>
<td>PHI B</td>
<td>00E3 R8</td>
</tr>
<tr>
<td>001D AB</td>
<td>PHI B</td>
<td>00E3 R8</td>
</tr>
<tr>
<td>001E F8 E3</td>
<td>LDI E3</td>
<td></td>
</tr>
<tr>
<td>0020 AB</td>
<td>PLO 8</td>
<td></td>
</tr>
<tr>
<td>0021 F8 F0</td>
<td>LDI F0</td>
<td></td>
</tr>
<tr>
<td>0023 A7</td>
<td>PLO 7</td>
<td></td>
</tr>
<tr>
<td>0024 F8 07</td>
<td>LDI 13</td>
<td></td>
</tr>
<tr>
<td>0025 B2</td>
<td>PHI 7</td>
<td></td>
</tr>
<tr>
<td>0026 B7</td>
<td>PHI 7</td>
<td></td>
</tr>
<tr>
<td>002B E3</td>
<td>SEX 3</td>
<td></td>
</tr>
<tr>
<td>002C 70 73</td>
<td>RET 73</td>
<td></td>
</tr>
<tr>
<td>002E C4</td>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>002F C4</td>
<td>NOP</td>
<td></td>
</tr>
</tbody>
</table>

Add. Code | Label | Mnemonic | Comments |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 F8 00</td>
<td>START</td>
<td>LDI 00</td>
<td>Load 0007 in R3 for use as the program counter (RP).</td>
</tr>
<tr>
<td>0002 B3</td>
<td>PHI 3</td>
<td>LDI 07</td>
<td></td>
</tr>
<tr>
<td>0003 F8 07</td>
<td>PHI 3</td>
<td>LDI 07</td>
<td></td>
</tr>
<tr>
<td>0005 A3</td>
<td>PLO 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0006 D3</td>
<td>SEP 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0007 7A</td>
<td>REQ</td>
<td>If monitor restarted from</td>
<td></td>
</tr>
<tr>
<td>0008 F8 00</td>
<td>PHI 1</td>
<td>Initialize the registers</td>
<td></td>
</tr>
<tr>
<td>001A A1</td>
<td>PHI 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000D AC</td>
<td>PHI B</td>
<td>0000 R1</td>
<td></td>
</tr>
<tr>
<td>000E BB</td>
<td>PHI I</td>
<td>07E0 R2</td>
<td></td>
</tr>
<tr>
<td>000F F8 01</td>
<td>PHI 4</td>
<td>0013 R5</td>
<td></td>
</tr>
<tr>
<td>0011 A4</td>
<td>PHI 5</td>
<td>07F0 R7</td>
<td></td>
</tr>
<tr>
<td>0012 B4</td>
<td>PHI A</td>
<td>00DC R8</td>
<td></td>
</tr>
<tr>
<td>0013 B5</td>
<td>PHI A</td>
<td>07E0 R2</td>
<td></td>
</tr>
<tr>
<td>0014 AA</td>
<td>PHI B</td>
<td>00E3 R8</td>
<td></td>
</tr>
<tr>
<td>0015 F8 02</td>
<td>PHI B</td>
<td>00E3 R8</td>
<td></td>
</tr>
<tr>
<td>0017 BA</td>
<td>PHI B</td>
<td>00E3 R8</td>
<td></td>
</tr>
<tr>
<td>0018 F8 E0</td>
<td>PHI B</td>
<td>00E3 R8</td>
<td></td>
</tr>
<tr>
<td>0019 A5</td>
<td>PHI B</td>
<td>00E3 R8</td>
<td></td>
</tr>
<tr>
<td>001B F8 DC</td>
<td>PHI B</td>
<td>00E3 R8</td>
<td></td>
</tr>
<tr>
<td>001C A8</td>
<td>PHI B</td>
<td>00E3 R8</td>
<td></td>
</tr>
<tr>
<td>001D AB</td>
<td>PHI B</td>
<td>00E3 R8</td>
<td></td>
</tr>
<tr>
<td>001E F8 E3</td>
<td>LDI E3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0020 AB</td>
<td>PLO 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0021 F8 F0</td>
<td>LDI F0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0023 A7</td>
<td>PLO 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0024 F8 07</td>
<td>LDI 13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0025 B2</td>
<td>PHI 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0026 B7</td>
<td>PHI 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>002B E3</td>
<td>SEX 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>002C 70 73</td>
<td>RET 73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>002E C4</td>
<td>NOP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>002F C4</td>
<td>NOP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Add. Code | Label | Mnemonic | Comments |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 F8 00</td>
<td>START</td>
<td>LDI 00</td>
<td>Load 0007 in R3 for use as the program counter (RP).</td>
</tr>
<tr>
<td>0002 B3</td>
<td>PHI 3</td>
<td>LDI 07</td>
<td></td>
</tr>
<tr>
<td>0003 F8 07</td>
<td>PHI 3</td>
<td>LDI 07</td>
<td></td>
</tr>
<tr>
<td>0005 A3</td>
<td>PLO 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0006 D3</td>
<td>SEP 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0007 7A</td>
<td>REQ</td>
<td>If monitor restarted from</td>
<td></td>
</tr>
<tr>
<td>0008 F8 00</td>
<td>PHI 1</td>
<td>Initialize the registers</td>
<td></td>
</tr>
<tr>
<td>001A A1</td>
<td>PHI 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000D AC</td>
<td>PHI B</td>
<td>0000 R1</td>
<td></td>
</tr>
<tr>
<td>000E BB</td>
<td>PHI I</td>
<td>07E0 R2</td>
<td></td>
</tr>
<tr>
<td>000F F8 01</td>
<td>PHI 4</td>
<td>0013 R5</td>
<td></td>
</tr>
<tr>
<td>0011 A4</td>
<td>PHI 5</td>
<td>07F0 R7</td>
<td></td>
</tr>
<tr>
<td>0012 B4</td>
<td>PHI A</td>
<td>00DC R8</td>
<td></td>
</tr>
<tr>
<td>0013 B5</td>
<td>PHI A</td>
<td>07E0 R2</td>
<td></td>
</tr>
<tr>
<td>0014 AA</td>
<td>PHI B</td>
<td>00E3 R8</td>
<td></td>
</tr>
<tr>
<td>0015 F8 02</td>
<td>PHI B</td>
<td>00E3 R8</td>
<td></td>
</tr>
<tr>
<td>0017 BA</td>
<td>PHI B</td>
<td>00E3 R8</td>
<td></td>
</tr>
<tr>
<td>0018 F8 E0</td>
<td>PHI B</td>
<td>00E3 R8</td>
<td></td>
</tr>
<tr>
<td>0019 A5</td>
<td>PHI B</td>
<td>00E3 R8</td>
<td></td>
</tr>
<tr>
<td>001B F8 DC</td>
<td>PHI B</td>
<td>00E3 R8</td>
<td></td>
</tr>
<tr>
<td>001C A8</td>
<td>PHI B</td>
<td>00E3 R8</td>
<td></td>
</tr>
<tr>
<td>001D AB</td>
<td>PHI B</td>
<td>00E3 R8</td>
<td></td>
</tr>
<tr>
<td>001E F8 E3</td>
<td>LDI E3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0020 AB</td>
<td>PLO 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0021 F8 F0</td>
<td>LDI F0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0023 A7</td>
<td>PLO 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0024 F8 07</td>
<td>LDI 13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0025 B2</td>
<td>PHI 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0026 B7</td>
<td>PHI 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>002B E3</td>
<td>SEX 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>002C 70 73</td>
<td>RET 73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>002E C4</td>
<td>NOP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>002F C4</td>
<td>NOP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### MONITOR PROGRAM

<table>
<thead>
<tr>
<th>Add. Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0030 39 33</td>
<td>CHECK BNQ SIN</td>
<td>;Jmp 1 if Q=0, else return</td>
<td></td>
</tr>
<tr>
<td>0032 D5</td>
<td>RETURN</td>
<td>;to calling program</td>
<td></td>
</tr>
<tr>
<td>0033 D4 00F4</td>
<td>SIN SUB INPUT</td>
<td>;Get input from device</td>
<td></td>
</tr>
<tr>
<td>0035 9F</td>
<td>DIN CHI F</td>
<td>;Input into D</td>
<td></td>
</tr>
<tr>
<td>0037 FF 20</td>
<td>SMI 20</td>
<td>;Subtract 20, 0 if &quot;space&quot;</td>
<td></td>
</tr>
<tr>
<td>0039 3A 3E</td>
<td>BNZ S1</td>
<td>;Jmp if not</td>
<td></td>
</tr>
<tr>
<td>003B 1C</td>
<td>INC C</td>
<td>;Make RC.0 &gt; 0, data to RE</td>
<td></td>
</tr>
<tr>
<td>003C 30 30</td>
<td>BR CHECK</td>
<td>;Back for more input</td>
<td></td>
</tr>
<tr>
<td>003E FF 01 S1</td>
<td>SMI 01</td>
<td>;Subtract 01, 0 if &quot;!&quot;</td>
<td></td>
</tr>
<tr>
<td>0040 32 C0</td>
<td>BZ MOD</td>
<td>;Jmp to modify memory if so</td>
<td></td>
</tr>
<tr>
<td>0042 FF 03</td>
<td>SMI 03</td>
<td>;Subtract 03, 0 if &quot;$&quot;</td>
<td></td>
</tr>
<tr>
<td>0044 3A 4B</td>
<td>BNZ S1B</td>
<td>;Jmp if not</td>
<td></td>
</tr>
<tr>
<td>0046 8D</td>
<td>CLO D</td>
<td>;Put RD into R0, and begin</td>
<td></td>
</tr>
<tr>
<td>0047 A0</td>
<td>PLO 0</td>
<td>;execution of a new program</td>
<td></td>
</tr>
<tr>
<td>0048 9D</td>
<td>CHI D</td>
<td>;with R0 as the PC. The PC</td>
<td></td>
</tr>
<tr>
<td>0049 B0</td>
<td>PHI 0</td>
<td>;should be changed back to</td>
<td></td>
</tr>
<tr>
<td>004A D0</td>
<td>SEP 0</td>
<td>;R3 by the new program.</td>
<td></td>
</tr>
<tr>
<td>004B FF 1B S1B</td>
<td>SMI 1B</td>
<td>;Subtract 1B, 0 if &quot;?&quot;,</td>
<td></td>
</tr>
<tr>
<td>0053 B3 74</td>
<td>BM NUM</td>
<td>;If neg, was 0 to 9, input it</td>
<td></td>
</tr>
<tr>
<td>0055 C2 0150</td>
<td>LBZ GET</td>
<td>;Jmp to GET if &quot;C&quot;</td>
<td></td>
</tr>
<tr>
<td>0058 FF 09</td>
<td>SMI 09</td>
<td>;Subtract 09, 0 if &quot;P&quot;</td>
<td></td>
</tr>
<tr>
<td>005A C2 0134</td>
<td>LBZ PUSH</td>
<td>;Jmp to PUSH if &quot;P&quot;</td>
<td></td>
</tr>
<tr>
<td>005D FF 02</td>
<td>SMI 02</td>
<td>;Subtract 02, 0 if &quot;R&quot;</td>
<td></td>
</tr>
<tr>
<td>005F C5</td>
<td>LSNZ</td>
<td>;Skip 2 if not</td>
<td></td>
</tr>
<tr>
<td>0060 D5</td>
<td>RETURN</td>
<td>;Return to the program</td>
<td></td>
</tr>
<tr>
<td>0061 00</td>
<td>IDL</td>
<td>;which called the monitor</td>
<td></td>
</tr>
<tr>
<td>0062 FF 02</td>
<td>SMI 02</td>
<td>;Subtract 02, 0 if &quot;T&quot;</td>
<td></td>
</tr>
<tr>
<td>0064 C2 0178</td>
<td>LBZ TTY</td>
<td>;Jmp to TTY routines if &quot;T&quot;</td>
<td></td>
</tr>
<tr>
<td>0067 FF 03</td>
<td>SMI 03</td>
<td>;Subtract 03, 0 if &quot;W&quot;</td>
<td></td>
</tr>
<tr>
<td>0069 C2 0120</td>
<td>LBZ WRITE</td>
<td>;Jmp to write if &quot;W&quot;</td>
<td></td>
</tr>
<tr>
<td>006C D1</td>
<td>SEP 1</td>
<td>;If no command recognized,</td>
<td></td>
</tr>
<tr>
<td>006D 00</td>
<td>IDL</td>
<td>;there was an error, so</td>
<td></td>
</tr>
<tr>
<td>006E 00</td>
<td>IDL</td>
<td>;restart monitor,</td>
<td></td>
</tr>
<tr>
<td>006F 00</td>
<td>IDL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### MONITOR PROGRAM

<table>
<thead>
<tr>
<th>Add. Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0070 F8 09</td>
<td>ATOF</td>
<td>LDI 09</td>
<td>;Add 09 to the ASCII from A to F to remove offset</td>
</tr>
<tr>
<td>0072 F4</td>
<td></td>
<td>ADD</td>
<td>;Restore it like 0 to 9</td>
</tr>
<tr>
<td>0073 57</td>
<td></td>
<td>STR 7</td>
<td>;last of offset. There is now a hex number in D4 to D7</td>
</tr>
<tr>
<td>0074 FO</td>
<td></td>
<td>NUM LDX</td>
<td>;of the D register.</td>
</tr>
<tr>
<td>0075 FE</td>
<td></td>
<td>SHL</td>
<td>;Shift left by 4 to remove</td>
</tr>
<tr>
<td>0076 FE</td>
<td></td>
<td>SHL SHL SHL SHL</td>
<td></td>
</tr>
<tr>
<td>0077 FE</td>
<td></td>
<td>SHL</td>
<td>;Save it</td>
</tr>
<tr>
<td>0078 FE</td>
<td></td>
<td>STR 7</td>
<td></td>
</tr>
<tr>
<td>0079 57</td>
<td></td>
<td>LDI 04</td>
<td>;Load R9,0 with the number of shifts to be done.</td>
</tr>
<tr>
<td>007A F8 04</td>
<td>MOSH</td>
<td>GLO C</td>
<td>;If RC,0=0, shift hex to RD</td>
</tr>
<tr>
<td>007B 3A 8D</td>
<td></td>
<td>BNZ LRE</td>
<td>;Else jmp to load RE</td>
</tr>
<tr>
<td>0080 8D</td>
<td></td>
<td>CLO D</td>
<td>;This section of code shifts RD left by one in a 16 bit</td>
</tr>
<tr>
<td>0081 FE</td>
<td></td>
<td>SHL</td>
<td>;shift. This leaves a 0 in the lsb.</td>
</tr>
<tr>
<td>0082 AD</td>
<td></td>
<td>PLO D</td>
<td></td>
</tr>
<tr>
<td>0083 9D</td>
<td></td>
<td>GHI D</td>
<td></td>
</tr>
<tr>
<td>0084 7E</td>
<td></td>
<td>SHLC</td>
<td></td>
</tr>
<tr>
<td>0085 BD</td>
<td></td>
<td>PHI D</td>
<td>;Shift done here</td>
</tr>
<tr>
<td>0086 FO</td>
<td></td>
<td>LDX</td>
<td>;Get the hex character</td>
</tr>
<tr>
<td>0087 FE</td>
<td></td>
<td>SHL</td>
<td>;Shift the msb into DF</td>
</tr>
<tr>
<td>0088 3B 98</td>
<td></td>
<td>BNF SHOV?</td>
<td>;If 0, check if shifting done</td>
</tr>
<tr>
<td>0089 1D</td>
<td></td>
<td>INC D</td>
<td>;Else inc RD, ie set lsb to 1</td>
</tr>
<tr>
<td>008A 30 98</td>
<td></td>
<td>BR SHOV?</td>
<td>;Check if shifting done</td>
</tr>
<tr>
<td>008B 8E</td>
<td></td>
<td>GLO E</td>
<td>;This is the same as 0080 to 008A, but the hex is shifted into RE because RC,0 &gt; 0.</td>
</tr>
<tr>
<td>008C 8E</td>
<td></td>
<td>SHL</td>
<td></td>
</tr>
<tr>
<td>008D AE</td>
<td></td>
<td>PLO E</td>
<td></td>
</tr>
<tr>
<td>008E 9E</td>
<td></td>
<td>GHI E</td>
<td></td>
</tr>
<tr>
<td>008F 7E</td>
<td></td>
<td>SHLC</td>
<td></td>
</tr>
<tr>
<td>0090 7E</td>
<td></td>
<td>PHI E</td>
<td></td>
</tr>
<tr>
<td>0092 FE</td>
<td></td>
<td>LDX</td>
<td></td>
</tr>
<tr>
<td>0093 FE</td>
<td></td>
<td>SHL</td>
<td></td>
</tr>
<tr>
<td>0094 3B 98</td>
<td></td>
<td>BNF SHOV?</td>
<td></td>
</tr>
<tr>
<td>0095 1E</td>
<td></td>
<td>INC RE</td>
<td></td>
</tr>
<tr>
<td>0096 57</td>
<td></td>
<td>STR 7</td>
<td>;Save the shifted hex</td>
</tr>
<tr>
<td>0097 29</td>
<td></td>
<td>DEC R9</td>
<td>;Dec the shift counter</td>
</tr>
<tr>
<td>0098 89</td>
<td></td>
<td>CLO 9</td>
<td>;Get the counter</td>
</tr>
<tr>
<td>0099 3A 7D</td>
<td></td>
<td>BNZ MOSH</td>
<td>;Jmp for more if not 0</td>
</tr>
<tr>
<td>009A D4 01 AB</td>
<td></td>
<td>SUB DISPY</td>
<td>;If done, display result, then back for more.</td>
</tr>
<tr>
<td>009B 30 30</td>
<td></td>
<td>BR CHECK</td>
<td></td>
</tr>
<tr>
<td>009C 00</td>
<td></td>
<td>IDL</td>
<td></td>
</tr>
<tr>
<td>009D 00</td>
<td></td>
<td>IDL</td>
<td></td>
</tr>
<tr>
<td>Add.</td>
<td>Code</td>
<td>Label</td>
<td>Mnemonic</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>-----------</td>
<td>----------</td>
</tr>
<tr>
<td>00A4</td>
<td>1D</td>
<td>LOGAN</td>
<td>INC D</td>
</tr>
<tr>
<td>00A5</td>
<td>0D</td>
<td>LOOK</td>
<td>LDN D</td>
</tr>
<tr>
<td>00A6</td>
<td>AE</td>
<td>PLO</td>
<td>E</td>
</tr>
<tr>
<td>00A7</td>
<td>F8 00</td>
<td>LDI 00</td>
<td>PHI E</td>
</tr>
<tr>
<td>00A8</td>
<td>D4 01AB</td>
<td>SUB DISPY</td>
<td>LSNQ</td>
</tr>
<tr>
<td>00A9</td>
<td>D5</td>
<td>RETURN</td>
<td></td>
</tr>
<tr>
<td>00A9</td>
<td>00</td>
<td>IDL</td>
<td></td>
</tr>
<tr>
<td>00B0</td>
<td>D4 00F4</td>
<td>SUB INPUT</td>
<td>GHI F</td>
</tr>
<tr>
<td>00B1</td>
<td>FF 49</td>
<td>SMI 49</td>
<td></td>
</tr>
<tr>
<td>00B2</td>
<td>32 A4</td>
<td>BZ LOGAN</td>
<td></td>
</tr>
<tr>
<td>00B3</td>
<td>F8 00</td>
<td>MOD</td>
<td>LDI 00</td>
</tr>
<tr>
<td>00B4</td>
<td>AC</td>
<td>GLO E</td>
<td></td>
</tr>
<tr>
<td>00B5</td>
<td>30 36</td>
<td>BR DIN</td>
<td>IDL</td>
</tr>
<tr>
<td>00B6</td>
<td>00</td>
<td>IDL</td>
<td>MOD</td>
</tr>
<tr>
<td>00B7</td>
<td>00</td>
<td>IDL</td>
<td></td>
</tr>
<tr>
<td>00B8</td>
<td>00</td>
<td>IDL</td>
<td>MOD</td>
</tr>
<tr>
<td>00B9</td>
<td>F8 00</td>
<td>MOD</td>
<td>LDI 00</td>
</tr>
<tr>
<td>00B4</td>
<td>BE</td>
<td>PHI E</td>
<td></td>
</tr>
<tr>
<td>00C9</td>
<td>D4 01AB</td>
<td>SUB DISPY</td>
<td>LSNQ</td>
</tr>
<tr>
<td>00C0</td>
<td>C5</td>
<td>RETURN</td>
<td></td>
</tr>
<tr>
<td>00C1</td>
<td>D5</td>
<td>IDL</td>
<td></td>
</tr>
<tr>
<td>00C2</td>
<td>00</td>
<td>IDL</td>
<td>SUB INPUT</td>
</tr>
<tr>
<td>00C3</td>
<td>9F</td>
<td>GHI F</td>
<td></td>
</tr>
<tr>
<td>00C4</td>
<td>FF 21</td>
<td>SMI 21</td>
<td></td>
</tr>
<tr>
<td>00C9</td>
<td>3A B8</td>
<td>BNZ RENM</td>
<td></td>
</tr>
<tr>
<td>00C9</td>
<td>D1</td>
<td>INC D</td>
<td></td>
</tr>
<tr>
<td>00C9</td>
<td>D4 01AB</td>
<td>SUB DISPY</td>
<td></td>
</tr>
<tr>
<td>00DD</td>
<td>30 30</td>
<td>BR CHECK</td>
<td>IDL</td>
</tr>
<tr>
<td>00DA</td>
<td>00</td>
<td>IDL</td>
<td></td>
</tr>
<tr>
<td>00DB</td>
<td>D3</td>
<td>SOVER</td>
<td>SEP 3</td>
</tr>
<tr>
<td>00DC</td>
<td>F6</td>
<td>SRX4</td>
<td>SHR</td>
</tr>
<tr>
<td>00DD</td>
<td>F6</td>
<td>SHR</td>
<td></td>
</tr>
<tr>
<td>00DE</td>
<td>F6</td>
<td>SHR</td>
<td></td>
</tr>
<tr>
<td>00DF</td>
<td>F6</td>
<td>SHR</td>
<td></td>
</tr>
<tr>
<td>00E0</td>
<td>30 30</td>
<td>BR SOVER</td>
<td></td>
</tr>
</tbody>
</table>
MONITOR PROGRAM

<table>
<thead>
<tr>
<th>Add.</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>00E2</td>
<td>D3</td>
<td>HOVER</td>
<td>SEP 3</td>
<td>Return to calling program</td>
</tr>
<tr>
<td>00E3</td>
<td>FA</td>
<td>0F</td>
<td>HXOUT</td>
<td>This is an SEP subroutine</td>
</tr>
<tr>
<td>00E5</td>
<td>FB</td>
<td>0F</td>
<td>XRI 0F</td>
<td>which outputs a single hex</td>
</tr>
<tr>
<td>00E7</td>
<td>57</td>
<td>STR 7</td>
<td></td>
<td>digit to the correct place</td>
</tr>
<tr>
<td>00E8</td>
<td>89</td>
<td>GLO 9</td>
<td></td>
<td>in the led display. Enter</td>
</tr>
<tr>
<td>00E9</td>
<td>FC</td>
<td>10</td>
<td>ADI 10</td>
<td>with the digit in D0 to D3</td>
</tr>
<tr>
<td>00EB</td>
<td>A9</td>
<td>PLO 9</td>
<td></td>
<td>of D, and the digit # minus</td>
</tr>
<tr>
<td>00EC</td>
<td>F1</td>
<td>OR</td>
<td></td>
<td>one in D4 to D7 of R9.0, so</td>
</tr>
<tr>
<td>00ED</td>
<td>57</td>
<td>STR 7</td>
<td></td>
<td>F0 is leftmost digit, and</td>
</tr>
<tr>
<td>00EE</td>
<td>63</td>
<td>OUT</td>
<td>DISP</td>
<td>60 is the rightmost digit.</td>
</tr>
<tr>
<td>00EF</td>
<td>27</td>
<td>DEC 7</td>
<td></td>
<td>This digit # is upped by 10.</td>
</tr>
<tr>
<td>00F0</td>
<td>30</td>
<td>E2</td>
<td>BR HOVER</td>
<td>Over and out. Note that this</td>
</tr>
<tr>
<td>00F2</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td>routine depends heavily on</td>
</tr>
<tr>
<td>00F3</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td>the display hardware.</td>
</tr>
</tbody>
</table>

;This is the input subroutine. It loops until it finds an input from the ASCII Keyboard or serial interface 1.
<table>
<thead>
<tr>
<th>Add.</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>00F4</td>
<td>3F</td>
<td>F9</td>
<td>INPUT</td>
<td>Jmp if no serial input</td>
</tr>
<tr>
<td>00F5</td>
<td>69</td>
<td>INPUT</td>
<td>F9 SER1</td>
<td>Else input it</td>
</tr>
<tr>
<td>00F7</td>
<td>30</td>
<td>FC</td>
<td>BR GOTIT</td>
<td>Jmp a bit</td>
</tr>
<tr>
<td>00F9</td>
<td>3F</td>
<td>F4</td>
<td>KB?</td>
<td>Loop if no keyboard either.</td>
</tr>
<tr>
<td>00FB</td>
<td>6B</td>
<td>INPUT</td>
<td>ASKEY</td>
<td>Else input it</td>
</tr>
<tr>
<td>00FC</td>
<td>BF</td>
<td>GOTIT</td>
<td>PHI F</td>
<td>Salt it away</td>
</tr>
<tr>
<td>00FD</td>
<td>D5</td>
<td>RETURN</td>
<td></td>
<td>Over and out</td>
</tr>
<tr>
<td>00FE</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00FF</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

;This is the SUB subroutine. It is used to call subroutines via the RCA SCRT (Standard Call and Return) conventions.
<table>
<thead>
<tr>
<th>Add.</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100</td>
<td>D3</td>
<td>EXITC</td>
<td>SEP 3</td>
<td>Go to the called subroutine.</td>
</tr>
<tr>
<td>0101</td>
<td>E2</td>
<td>SUB</td>
<td>SEX 2</td>
<td>R2 points to the return</td>
</tr>
<tr>
<td>0102</td>
<td>96</td>
<td>CHI</td>
<td>6</td>
<td>address stack.</td>
</tr>
<tr>
<td>0103</td>
<td>73</td>
<td>STXD</td>
<td></td>
<td>Push R6 onto stack and</td>
</tr>
<tr>
<td>0104</td>
<td>86</td>
<td>CLO</td>
<td>6</td>
<td>leave stack at a free</td>
</tr>
<tr>
<td>0105</td>
<td>73</td>
<td>STXD</td>
<td></td>
<td>location.</td>
</tr>
<tr>
<td>0106</td>
<td>93</td>
<td>CHI</td>
<td>3</td>
<td>Copy old RP (R3) into R6</td>
</tr>
<tr>
<td>0107</td>
<td>B6</td>
<td>PHI</td>
<td>6</td>
<td>to save it.</td>
</tr>
<tr>
<td>0108</td>
<td>83</td>
<td>GLO</td>
<td>3</td>
<td>R6 now points to two byte</td>
</tr>
<tr>
<td>0109</td>
<td>A6</td>
<td>PLO</td>
<td>6</td>
<td>inline address in calling</td>
</tr>
<tr>
<td>010A</td>
<td>46</td>
<td>LDA</td>
<td>6</td>
<td>program. Get it and put it</td>
</tr>
<tr>
<td>010B</td>
<td>B3</td>
<td>CHI</td>
<td>6</td>
<td>into R3 as it is the start</td>
</tr>
<tr>
<td>010C</td>
<td>46</td>
<td>LDA</td>
<td>6</td>
<td>of the called routine.</td>
</tr>
<tr>
<td>010D</td>
<td>A3</td>
<td>PLO</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>010E</td>
<td>E7</td>
<td>SEX</td>
<td>7</td>
<td>Put R7 back as RX as R2 only</td>
</tr>
<tr>
<td>010F</td>
<td>30</td>
<td>00</td>
<td>BR EXITC</td>
<td>points to the subroutine</td>
</tr>
<tr>
<td>0111</td>
<td>C0</td>
<td>IDL</td>
<td></td>
<td>address stack.</td>
</tr>
</tbody>
</table>


This is the RETURN subroutine. It is used to return from a subroutine called by standard RCA SCRT techniques. This and the above routines are not used for SEP subroutines.

<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0112</td>
<td>D3</td>
<td>EXITR</td>
<td>D3</td>
<td>Return to calling program.</td>
</tr>
<tr>
<td>0113</td>
<td>96</td>
<td>RETRN</td>
<td>GHI 6</td>
<td>Copy R6 into R3 so it will</td>
</tr>
<tr>
<td>0114</td>
<td>B3</td>
<td>PHI</td>
<td>3</td>
<td>Contain the return program</td>
</tr>
<tr>
<td>0115</td>
<td>86</td>
<td>GLO</td>
<td>6</td>
<td>Counter.</td>
</tr>
<tr>
<td>0116</td>
<td>A3</td>
<td>PLO</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>0117</td>
<td>E2</td>
<td>SEX</td>
<td>2</td>
<td>R2 points to the return</td>
</tr>
<tr>
<td>0118</td>
<td>60</td>
<td>IRX</td>
<td></td>
<td>Address stack. Decrement</td>
</tr>
<tr>
<td>0119</td>
<td>72</td>
<td>LDXA</td>
<td></td>
<td>It from the free location so</td>
</tr>
<tr>
<td>011A</td>
<td>A6</td>
<td>PLO</td>
<td>6</td>
<td>It points to the previously</td>
</tr>
<tr>
<td>011B</td>
<td>F0</td>
<td>LDX</td>
<td></td>
<td>Stored value of R6 and put</td>
</tr>
<tr>
<td>011C</td>
<td>B6</td>
<td>PHI</td>
<td>6</td>
<td>It back into R6.</td>
</tr>
<tr>
<td>011D</td>
<td>E7</td>
<td>SEX</td>
<td>7</td>
<td>Put R7 back as RX.</td>
</tr>
<tr>
<td>011E</td>
<td>30</td>
<td>12</td>
<td>BR EXITR</td>
<td>Over and out</td>
</tr>
</tbody>
</table>

This is the "W" (Write) routine. It is used to dump memory locations via serial output port 1. The data is sent out as a stream of 8 bit bytes, but final format depends on the UART control switches. It is not converted to hex or ASCII, so this is the routine for writing to cassette. The data from M(RD) to M(RE) is sent out.

<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0120</td>
<td>ED</td>
<td>WRITE</td>
<td>SEX D</td>
<td>RD is used for output (RX).</td>
</tr>
<tr>
<td>0121</td>
<td>36</td>
<td>21</td>
<td>TFRE? B3 TFRE?</td>
<td>Loop 'till transmitter free.</td>
</tr>
<tr>
<td>0123</td>
<td>61</td>
<td>OUT</td>
<td>SER1</td>
<td>Output byte and advance RD.</td>
</tr>
<tr>
<td>0124</td>
<td>E7</td>
<td>SEX</td>
<td>7</td>
<td>R7 = RX for display.</td>
</tr>
<tr>
<td>0125</td>
<td>D4</td>
<td>01AB</td>
<td>SUB DISPY</td>
<td>Display advanced address.</td>
</tr>
<tr>
<td>0126</td>
<td>D4</td>
<td>01C4</td>
<td>SUB SRERD</td>
<td>Sub for 16 bit RE-RD and</td>
</tr>
<tr>
<td>0127</td>
<td>33</td>
<td>20</td>
<td>BGE WRITE</td>
<td>back for more if RE&gt;RD.</td>
</tr>
<tr>
<td>0128</td>
<td>C5</td>
<td>LSNQ</td>
<td></td>
<td>Skip 2 if Q=0, else this was</td>
</tr>
<tr>
<td>012E</td>
<td>D5</td>
<td>RETURN</td>
<td></td>
<td>Called as a subroutine, so</td>
</tr>
<tr>
<td>012F</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td>Return to calling program.</td>
</tr>
<tr>
<td>0130</td>
<td>D1</td>
<td>SEP</td>
<td>1</td>
<td>Done, so back to monitor.</td>
</tr>
<tr>
<td>0131</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0132</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0133</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### MONITOR PROGRAM

This is the "P" (Push) routine. All the data from M(RD) to M(RE) is pushed down by one location. This is useful for inserting an instruction, but all jumps must be corrected.

<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0134</td>
<td>4D</td>
<td>PUSH</td>
<td>LDA D</td>
<td>Get the first byte &amp; advance</td>
</tr>
<tr>
<td>0135</td>
<td>B9</td>
<td>PHI</td>
<td>9</td>
<td>Save it.</td>
</tr>
<tr>
<td>0136</td>
<td>OD</td>
<td>MPUSH</td>
<td>LDN D</td>
<td>Get the second byte.</td>
</tr>
<tr>
<td>0137</td>
<td>A9</td>
<td>PLO</td>
<td>9</td>
<td>Save that too.</td>
</tr>
<tr>
<td>0138</td>
<td>5D</td>
<td>STR</td>
<td>D</td>
<td>Get the first one again.</td>
</tr>
<tr>
<td>0139</td>
<td>89</td>
<td>GHI</td>
<td>9</td>
<td>Put it in new location.</td>
</tr>
<tr>
<td>013A</td>
<td>89</td>
<td>CLO</td>
<td>9</td>
<td>Get the second one again.</td>
</tr>
<tr>
<td>013B</td>
<td>B9</td>
<td>PHI</td>
<td>9</td>
<td>Save it as new first byte.</td>
</tr>
<tr>
<td>013C</td>
<td>D4</td>
<td>01AB</td>
<td>SUB DISPY</td>
<td>Display advanced address.</td>
</tr>
<tr>
<td>013D</td>
<td>D4</td>
<td>01C4</td>
<td>SUB SRERD</td>
<td>Sub for 16 bit RE-RD.</td>
</tr>
<tr>
<td>0142</td>
<td>3B</td>
<td>47</td>
<td>BL PDONE</td>
<td>Jump ahead if done</td>
</tr>
<tr>
<td>0144</td>
<td>1D</td>
<td>INC</td>
<td>D</td>
<td>Else increment RD</td>
</tr>
<tr>
<td>0145</td>
<td>30</td>
<td>36</td>
<td>BR MPUSH</td>
<td>and go back for more.</td>
</tr>
<tr>
<td>0147</td>
<td>C5</td>
<td>PDONE</td>
<td>LSNQ</td>
<td>Skp 2 if Q=0, else this was</td>
</tr>
<tr>
<td>0148</td>
<td>D5</td>
<td>RETURN</td>
<td></td>
<td>called as a subroutine, so</td>
</tr>
<tr>
<td>0149</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td>return to calling program.</td>
</tr>
<tr>
<td>014A</td>
<td>D1</td>
<td>SEP</td>
<td>1</td>
<td>Done, so back to monitor.</td>
</tr>
<tr>
<td>014B</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td>IDL from 014B to 014F inc.</td>
</tr>
</tbody>
</table>

This is the "G" (Get) routine. It is used to input a string of data bytes into memory from M(RD) to M(RE) from serial port 1. Parity depends on the front panel switches.

This is the routine for loading a cassette or paper tape.

<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0150</td>
<td>69</td>
<td>GET</td>
<td>INP SER1</td>
<td>Input to clear UART flags.</td>
</tr>
<tr>
<td>0151</td>
<td>F8</td>
<td>FF</td>
<td>LDI FF</td>
<td>Load FFFF into R9 for use as</td>
</tr>
<tr>
<td>0153</td>
<td>B9</td>
<td>PHI</td>
<td>9</td>
<td>a timing constant. A delay</td>
</tr>
<tr>
<td>0154</td>
<td>F8</td>
<td>FF</td>
<td>LDI FF</td>
<td>is used so the cassette</td>
</tr>
<tr>
<td>0156</td>
<td>A9</td>
<td>PLO</td>
<td>9</td>
<td>interface can stabilize.</td>
</tr>
<tr>
<td>0157</td>
<td>29</td>
<td>DELAY</td>
<td>DEC 9</td>
<td>Decrement timing constant.</td>
</tr>
<tr>
<td>0158</td>
<td>B9</td>
<td>GHI</td>
<td>9</td>
<td>Get the high 8 bits.</td>
</tr>
<tr>
<td>0159</td>
<td>C4</td>
<td>NOP</td>
<td></td>
<td>Extra delay.</td>
</tr>
<tr>
<td>015A</td>
<td>C4</td>
<td>NOP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>015B</td>
<td>3A</td>
<td>57</td>
<td>BNZ DELAY</td>
<td>Loop until R9.1 = 00.</td>
</tr>
<tr>
<td>015D</td>
<td>35</td>
<td>5D</td>
<td>SER?</td>
<td>Tiny loop until data appears</td>
</tr>
<tr>
<td>015F</td>
<td>34</td>
<td>50</td>
<td>B1 GET</td>
<td>Start again if error.</td>
</tr>
<tr>
<td>0161</td>
<td>69</td>
<td>GMORE</td>
<td>INP SER1</td>
<td>Get some data (finally!!)</td>
</tr>
<tr>
<td>0162</td>
<td>5D</td>
<td>STR</td>
<td>D</td>
<td>and store it.</td>
</tr>
<tr>
<td>0163</td>
<td>1D</td>
<td>INC</td>
<td>D</td>
<td>Increment memory pointer.</td>
</tr>
<tr>
<td>0164</td>
<td>D4</td>
<td>01AB</td>
<td>SUB DISPY</td>
<td>Display pointer.</td>
</tr>
<tr>
<td>0167</td>
<td>D4</td>
<td>01C4</td>
<td>SUB SRERD</td>
<td>Sub for 16 bit RE - RD</td>
</tr>
<tr>
<td>016A</td>
<td>3B</td>
<td>70</td>
<td>BL GDONE</td>
<td>and jump ahead if done.</td>
</tr>
<tr>
<td>016E</td>
<td>3C</td>
<td>61</td>
<td>BN1 GMORE</td>
<td>Back for more if no error.</td>
</tr>
<tr>
<td>0170</td>
<td>C5</td>
<td>GDONE</td>
<td>LSNQ</td>
<td>Skp 2 if Q=0, else this was</td>
</tr>
<tr>
<td>0171</td>
<td>D5</td>
<td>RETURN</td>
<td></td>
<td>called as a subroutine, so</td>
</tr>
<tr>
<td>0172</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td>return to calling program.</td>
</tr>
<tr>
<td>0173</td>
<td>D1</td>
<td>SEP</td>
<td>1</td>
<td>All done, back to monitor</td>
</tr>
</tbody>
</table>
### MONITOR PROGRAM

These are the two teletype routines, "TT" and "TP".

**TT** takes sequential 8 bit memory locations and outputs
them as two hex ASCII characters - so 11000101 would
be sent out as C5. Each pair of characters is followed by
a line feed/carriage return. **TP** is similar except that the
starting address and every nn00 address is also output.

These routines are used for teletype listings. The memory
from M<RD> to M<RE> is output.

<table>
<thead>
<tr>
<th>Add.</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0174</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0175</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0176</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0177</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0178</td>
<td>D4</td>
<td>00F4</td>
<td>TTY SUB</td>
<td>INPUT</td>
</tr>
<tr>
<td>0179</td>
<td>9F</td>
<td>GHI F</td>
<td></td>
<td>result in RC, 1 for flag use.</td>
</tr>
<tr>
<td>017C</td>
<td>FF</td>
<td>50</td>
<td>SMI 50</td>
<td></td>
</tr>
<tr>
<td>017E</td>
<td>BC</td>
<td>PHI C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>017F</td>
<td>D4</td>
<td>01DA</td>
<td>SUB CRLF</td>
<td></td>
</tr>
<tr>
<td>0182</td>
<td>9C</td>
<td>GHI C</td>
<td></td>
<td>Get flag and</td>
</tr>
<tr>
<td>0183</td>
<td>3A</td>
<td>8B</td>
<td>NOTTP</td>
<td></td>
</tr>
<tr>
<td>0185</td>
<td>D4</td>
<td>01CE</td>
<td>RDOUT</td>
<td>Output RD to serial port 1.</td>
</tr>
<tr>
<td>0188</td>
<td>D4</td>
<td>01DA</td>
<td>MORTY</td>
<td>SUB CRLF</td>
</tr>
<tr>
<td>018B</td>
<td>4D</td>
<td>NOTTP</td>
<td>LDA D</td>
<td>Get the data, put it in RF.1</td>
</tr>
<tr>
<td>018C</td>
<td>BF</td>
<td>PHI F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>018D</td>
<td>D4</td>
<td>01E2</td>
<td>2HOUT</td>
<td>Sub to output two characters</td>
</tr>
<tr>
<td>0190</td>
<td>D4</td>
<td>01AB</td>
<td>DISPY</td>
<td>Display advanced pointer.</td>
</tr>
<tr>
<td>0193</td>
<td>9C</td>
<td>GHI C</td>
<td></td>
<td>Get flag and</td>
</tr>
<tr>
<td>0194</td>
<td>3A</td>
<td>A1</td>
<td>NOADD</td>
<td>don't output add, if not TP.</td>
</tr>
<tr>
<td>0196</td>
<td>8D</td>
<td>GLO D</td>
<td></td>
<td>Get low 8 bits of pointer,</td>
</tr>
<tr>
<td>0197</td>
<td>FA</td>
<td>0F</td>
<td>ANI 0F</td>
<td>mask in lower 4 bits only.</td>
</tr>
<tr>
<td>0199</td>
<td>3A</td>
<td>A1</td>
<td>NOADD</td>
<td>Jump if address not = nn00,</td>
</tr>
<tr>
<td>019B</td>
<td>D4</td>
<td>01DA</td>
<td>CRLF</td>
<td>Output a CR and LF.</td>
</tr>
<tr>
<td>019E</td>
<td>D4</td>
<td>01CE</td>
<td>RDOUT</td>
<td>Output RD.</td>
</tr>
<tr>
<td>01A1</td>
<td>D4</td>
<td>01C4</td>
<td>NOADD</td>
<td>SUB SRERD; Sub for 16 bit RE - RD.</td>
</tr>
<tr>
<td>01A4</td>
<td>33</td>
<td>BBE</td>
<td>MORTY</td>
<td>Go back, for more if not done</td>
</tr>
<tr>
<td>01A6</td>
<td>C5</td>
<td>LSNQ</td>
<td></td>
<td>Skp 2 if Q=0, else this was</td>
</tr>
<tr>
<td>01A7</td>
<td>D5</td>
<td>RETURN</td>
<td></td>
<td>called as a subroutine, so</td>
</tr>
<tr>
<td>01A8</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td>retron to calling program.</td>
</tr>
<tr>
<td>01A9</td>
<td>D1</td>
<td>SEP 1</td>
<td></td>
<td>A1 done, back to monitor</td>
</tr>
<tr>
<td>01AA</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### MONITOR PROGRAM

This is the display subroutine. When it is called the current contents of RD and RE are output to an eight digit hexadecimal led display. The display is self scanning and refresh by the program is not required. The display should be updated whenever RD or RE is changed.

<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01AB</td>
<td>F8</td>
<td>DISPY</td>
<td>LDI FO</td>
<td>Preset R9.0 for use by HXOUT</td>
</tr>
<tr>
<td>01AD</td>
<td>A9</td>
<td>DISPY</td>
<td>LDI FO</td>
<td></td>
</tr>
<tr>
<td>01AE</td>
<td>9D</td>
<td>CHI</td>
<td>D</td>
<td>Get the high 8 bits of RD.</td>
</tr>
<tr>
<td>01AF</td>
<td>DB</td>
<td>SEP</td>
<td>8</td>
<td>Sub to shift right by four.</td>
</tr>
<tr>
<td>01B0</td>
<td>DB</td>
<td>SEP</td>
<td>B</td>
<td>Sub to output first (left-most) digit, get RD.1 again.</td>
</tr>
<tr>
<td>01B1</td>
<td>9D</td>
<td>CHI</td>
<td>D</td>
<td>Get RD.1 again.</td>
</tr>
<tr>
<td>01B2</td>
<td>DB</td>
<td>SEP</td>
<td>B</td>
<td>Output second digit.</td>
</tr>
<tr>
<td>01B3</td>
<td>8D</td>
<td>GLO</td>
<td>D</td>
<td>Get the low 8 bits of RD.</td>
</tr>
<tr>
<td>01B4</td>
<td>DB</td>
<td>SEP</td>
<td>8</td>
<td>Shift right by four.</td>
</tr>
<tr>
<td>01B5</td>
<td>DB</td>
<td>SEP</td>
<td>B</td>
<td>Output third digit.</td>
</tr>
<tr>
<td>01B6</td>
<td>8D</td>
<td>GLO</td>
<td>8</td>
<td>Get RD.0 again.</td>
</tr>
<tr>
<td>01B7</td>
<td>DB</td>
<td>SEP</td>
<td>B</td>
<td>Output fourth digit.</td>
</tr>
<tr>
<td>01B8</td>
<td>9E</td>
<td>CHI</td>
<td>E</td>
<td>Get the high 8 bits of RE.</td>
</tr>
<tr>
<td>01B9</td>
<td>DB</td>
<td>SEP</td>
<td>8</td>
<td>Shift right by four.</td>
</tr>
<tr>
<td>01BA</td>
<td>DB</td>
<td>SEP</td>
<td>B</td>
<td>Output fifth digit.</td>
</tr>
<tr>
<td>01BB</td>
<td>9E</td>
<td>CHI</td>
<td>E</td>
<td>Get RE.1 again.</td>
</tr>
<tr>
<td>01BC</td>
<td>DB</td>
<td>SEP</td>
<td>B</td>
<td>Output sixth digit.</td>
</tr>
<tr>
<td>01BD</td>
<td>8E</td>
<td>GLO</td>
<td>E</td>
<td>Get the low 8 bits of RE.</td>
</tr>
<tr>
<td>01BE</td>
<td>DB</td>
<td>SEP</td>
<td>8</td>
<td>Shift right by four.</td>
</tr>
<tr>
<td>01BF</td>
<td>DB</td>
<td>SEP</td>
<td>B</td>
<td>Output seventh digit.</td>
</tr>
<tr>
<td>01C0</td>
<td>8E</td>
<td>GLO</td>
<td>E</td>
<td>Get RE.0 again.</td>
</tr>
<tr>
<td>01C1</td>
<td>DB</td>
<td>SEP</td>
<td>B</td>
<td>Output eighth digit.</td>
</tr>
<tr>
<td>01C2</td>
<td>D5</td>
<td>RETURN</td>
<td></td>
<td>Return to calling program.</td>
</tr>
<tr>
<td>01C3</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This routine performs a 16 bit subtraction, RE - RD. Only the sign, in DF, is returned.

<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01C4</td>
<td>8D</td>
<td>SRERD</td>
<td>GLO D</td>
<td>Get low 8 bits of RD and put it in a free location.</td>
</tr>
<tr>
<td>01C5</td>
<td>57</td>
<td>STR</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>01C6</td>
<td>8E</td>
<td>GLO</td>
<td>E</td>
<td>Get low 8 bits of RE.</td>
</tr>
<tr>
<td>01C7</td>
<td>F7</td>
<td>SM</td>
<td></td>
<td>D-RX = RE.0-RD.0.</td>
</tr>
<tr>
<td>01C8</td>
<td>9D</td>
<td>CHI</td>
<td>D</td>
<td>Get high 8 bits of RD and put it in a free location.</td>
</tr>
<tr>
<td>01C9</td>
<td>57</td>
<td>STR</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>01CA</td>
<td>9E</td>
<td>GHI</td>
<td>E</td>
<td>Get high 8 bits of RE.</td>
</tr>
<tr>
<td>01CB</td>
<td>77</td>
<td>SMB</td>
<td></td>
<td>RE.1 - RD.1 - Borrow.</td>
</tr>
<tr>
<td>01CC</td>
<td>D5</td>
<td>RETURN</td>
<td></td>
<td>Return to calling program.</td>
</tr>
<tr>
<td>01CD</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>Code</td>
<td>Label</td>
<td>Mnemonic</td>
<td>Comments</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td>-------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>01CE</td>
<td>9D</td>
<td>RDOT</td>
<td>GHI</td>
<td>Get the high 8 bits of RD</td>
</tr>
<tr>
<td>01CF</td>
<td>BF</td>
<td></td>
<td>PHI</td>
<td>;and put them in RF,1.</td>
</tr>
<tr>
<td>01D0</td>
<td>D4</td>
<td>01E2</td>
<td>SUB 2HOUT</td>
<td>;Sub to output two characters</td>
</tr>
<tr>
<td>01D3</td>
<td>BD</td>
<td>GLO</td>
<td>D</td>
<td>Get the low 8 bits of RD</td>
</tr>
<tr>
<td>01D4</td>
<td>BF</td>
<td>PHI</td>
<td>F</td>
<td>;and put them in RF,1.</td>
</tr>
<tr>
<td>01D5</td>
<td>D4</td>
<td>01E2</td>
<td>SUB 1HOUT</td>
<td>;Sub to output two characters</td>
</tr>
<tr>
<td>01D8</td>
<td>D5</td>
<td>RETURN</td>
<td></td>
<td>;Return to calling program.</td>
</tr>
<tr>
<td>01D9</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

;This routine outputs a "CARRIAGE RETURN" & "LINEFEED".

<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01DA</td>
<td>F8</td>
<td>0D</td>
<td>CRLF</td>
<td>LDI</td>
</tr>
<tr>
<td>01DC</td>
<td>DA</td>
<td></td>
<td>SEP</td>
<td>A</td>
</tr>
<tr>
<td>01DD</td>
<td>F8</td>
<td>0A</td>
<td>LDI</td>
<td>OA</td>
</tr>
<tr>
<td>01DF</td>
<td>DA</td>
<td></td>
<td>SEP</td>
<td>A</td>
</tr>
<tr>
<td>01E0</td>
<td>D5</td>
<td>RETURN</td>
<td></td>
<td>;Return to calling program.</td>
</tr>
<tr>
<td>01E1</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

;This routine outputs two ASCII hexadecimal characters from
;an 8 bit number in RF,1.

<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01E2</td>
<td>9F</td>
<td>2HOUT</td>
<td>GHI</td>
<td>Get data passed from program</td>
</tr>
<tr>
<td>01E3</td>
<td>B9</td>
<td>PHI</td>
<td>9</td>
<td>;Hide it somewhere,</td>
</tr>
<tr>
<td>01E4</td>
<td>D8</td>
<td>SEP</td>
<td>8</td>
<td>;Sub to shift right by four.</td>
</tr>
<tr>
<td>01E5</td>
<td>BF</td>
<td>PHI</td>
<td>F</td>
<td>;Put it in RF,1.</td>
</tr>
<tr>
<td>01E6</td>
<td>D4</td>
<td>01F2</td>
<td>SUB 1HOUT</td>
<td>;Sub to output one character.</td>
</tr>
<tr>
<td>01E9</td>
<td>99</td>
<td>GHI</td>
<td>9</td>
<td>;Get original data again and</td>
</tr>
<tr>
<td>01EA</td>
<td>FA</td>
<td>0F</td>
<td>ANI</td>
<td>OF</td>
</tr>
<tr>
<td>01EC</td>
<td>BF</td>
<td>PHI</td>
<td>F</td>
<td>;Put it in RF,1.</td>
</tr>
<tr>
<td>01ED</td>
<td>D4</td>
<td>01F2</td>
<td>SUB 1HOUT</td>
<td>;Sub to output one character.</td>
</tr>
<tr>
<td>01F0</td>
<td>D5</td>
<td>RETURN</td>
<td></td>
<td>;Return to calling program.</td>
</tr>
<tr>
<td>01F1</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

;This routine outputs one ASCII hex character from the
;lower four bits of RF,1.

<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01F2</td>
<td>9F</td>
<td>1HOUT</td>
<td>GHI</td>
<td>Get data passed from program</td>
</tr>
<tr>
<td>01F3</td>
<td>FF</td>
<td>SMI</td>
<td>0A</td>
<td>;Subtract 0A, &lt;0 if 0 to 9,</td>
</tr>
<tr>
<td>01F5</td>
<td>9F</td>
<td>GHI</td>
<td>F</td>
<td>;Get data again, but leave DF</td>
</tr>
<tr>
<td>01F6</td>
<td>C7</td>
<td>LSNF</td>
<td></td>
<td>;Skp 2 if neg, was 0 to 9.</td>
</tr>
<tr>
<td>01F7</td>
<td>FC</td>
<td>07</td>
<td>ADI</td>
<td>07</td>
</tr>
<tr>
<td>01F9</td>
<td>FC</td>
<td>30</td>
<td>ADI</td>
<td></td>
</tr>
<tr>
<td>01FB</td>
<td>DA</td>
<td>SEP</td>
<td>A</td>
<td>;Sub to output D.</td>
</tr>
<tr>
<td>01FC</td>
<td>D5</td>
<td>RETURN</td>
<td></td>
<td>;Return to calling program.</td>
</tr>
</tbody>
</table>
MONITOR PROGRAM

;This is the subroutine to output the contents of D to serial port 1. It does not modify any registers, but does require RX to point to a free memory location.

Add. Code Label Mnemonic Comments
01FD 00 IDL ;
01FE 00 IDL ;Note that RX must be R7 on entry.
01FF 00 IDL ;
0200 D3 SIDON SEP 3 ;Return to calling program.
0201 57 SOUT1 STR 7 ;Put D into memory.
0202 36 02 S10K? B3 S10K? ;Loop 'till transmitter free.
0204 61 OUT SER1 ;Output data.
0205 27 DEC 7 ;Restore RX value.
0206 30 00 BR SIDON ;Done, so get out.
0208 00 IDL ;IDL's (00) until 020F.

;This is the routine to output the contents of D to serial port 2. If RA is set to 0210 this routine will be used instead of SOUT1 whenever an SEP A (DA) is executed. This port is harder to use as the status must be read in with an input instruction and cannot be checked with the flags. It is more flexible as the control register is set with software rather than switches, but because of this the control register must be set before calling this program. This routine is not used by the monitor. RX must be R7 and point to 07F0 on entry. Location 07FF must be free.

020F D3 S2DON SEP 3 ;Return to calling program.
0210 57 SOUT2 STR 7 ;Save D in 07F0.
0211 F8 FF LDI FF ;Point to a free location.
0213 A7 PLO 7 ;
0214 6F T20K? INP PIP1 ;Input parallel port 1.
0215 FE SHL ;Put transmitter status bit into DF, loop until ready.
0216 3B 14 BNF T20K? ;Point to stored D in normal
0218 F8 F0 LDI F0 ;I/O location and
021A A7 PLO 7 ;output data.
021B 66 OUT SER2 ;
021C 27 DEC 7 ;Restore RX value and
021D 30 0F BR S2DON ;get out.
021F 00 IDL ;
APPENDIX D

MORSE DECODING PROGRAM
MORSE PROGRAM

;This is the Morse code decoding program. Program segments are more in logical than numerical order.

<table>
<thead>
<tr>
<th>Add. Code</th>
<th>Label Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>03DE</td>
<td>F8 00</td>
<td>RSTRT LDI 00</td>
</tr>
<tr>
<td>03DF</td>
<td>AC</td>
<td>PLO C</td>
</tr>
<tr>
<td>03E0</td>
<td>F8 03</td>
<td>MORSE LDI 03</td>
</tr>
<tr>
<td>03E2</td>
<td>B3</td>
<td>PHI 3</td>
</tr>
<tr>
<td>03E3</td>
<td>F8 E6</td>
<td>LDI E6</td>
</tr>
<tr>
<td>03E5</td>
<td>A3</td>
<td>PLO 3</td>
</tr>
<tr>
<td>03E6</td>
<td>D3</td>
<td>SEP 3</td>
</tr>
<tr>
<td>03E7</td>
<td>E3</td>
<td>SEX 3</td>
</tr>
<tr>
<td>03E8</td>
<td>67</td>
<td>OUT CON2</td>
</tr>
<tr>
<td>03E9</td>
<td>F0</td>
<td>(data)</td>
</tr>
<tr>
<td>03EA</td>
<td>E7</td>
<td>SEX 7</td>
</tr>
<tr>
<td>03EB</td>
<td>7A</td>
<td>REQ</td>
</tr>
<tr>
<td>03EC</td>
<td>D4 0033</td>
<td>SUB SIN</td>
</tr>
<tr>
<td>03EF</td>
<td>F8 10</td>
<td>RSTR2 LDI 10</td>
</tr>
<tr>
<td>03F1</td>
<td>AE</td>
<td>PLO E</td>
</tr>
<tr>
<td>03F2</td>
<td>BE</td>
<td>PHI E</td>
</tr>
<tr>
<td>03F3</td>
<td>D4 03C0</td>
<td>SUB MASF</td>
</tr>
<tr>
<td>03F6</td>
<td>D4 01DA RSTR1</td>
<td>SUB CRLF</td>
</tr>
<tr>
<td>03F9</td>
<td>F8 00</td>
<td>LDI 00</td>
</tr>
<tr>
<td>03FB</td>
<td>AC</td>
<td>PLO C</td>
</tr>
<tr>
<td>03FC</td>
<td>EC</td>
<td>PHI C</td>
</tr>
<tr>
<td>03FD</td>
<td>A9</td>
<td>PLO 9</td>
</tr>
<tr>
<td>03FE</td>
<td>B9</td>
<td>PHI 9</td>
</tr>
<tr>
<td>03FF</td>
<td>7A</td>
<td>REQ</td>
</tr>
<tr>
<td>0400</td>
<td>9D</td>
<td>TIME GHI D</td>
</tr>
<tr>
<td>0401</td>
<td>BF</td>
<td>PHI F</td>
</tr>
<tr>
<td>0402</td>
<td>8D</td>
<td>GLO D</td>
</tr>
<tr>
<td>0403</td>
<td>AF</td>
<td>PLO F</td>
</tr>
<tr>
<td>0404</td>
<td>8F</td>
<td>TIM1 GLO F</td>
</tr>
<tr>
<td>0405</td>
<td>32 0A</td>
<td>EZ TIM2</td>
</tr>
<tr>
<td>0407</td>
<td>2F</td>
<td>TIM3 DEC F</td>
</tr>
<tr>
<td>0408</td>
<td>30 04</td>
<td>BR TIM1</td>
</tr>
<tr>
<td>040A</td>
<td>9F</td>
<td>TIM2 GHI 9</td>
</tr>
<tr>
<td>040B</td>
<td>3A 07</td>
<td>BNZ TIM3</td>
</tr>
<tr>
<td>040D</td>
<td>6F</td>
<td>INP PIP1</td>
</tr>
<tr>
<td>040E</td>
<td>64</td>
<td>OUT POP2</td>
</tr>
<tr>
<td>040F</td>
<td>27</td>
<td>DEC 7</td>
</tr>
</tbody>
</table>
### MORSE PROGRAM

<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0410</td>
<td>FA</td>
<td>01</td>
<td>ANI 01</td>
<td>;Mask in lsb, 0=up, 1=down.</td>
</tr>
<tr>
<td>0412</td>
<td>32</td>
<td>2C</td>
<td>BZ UP</td>
<td>;Jump if key up.</td>
</tr>
<tr>
<td>0414</td>
<td>8C</td>
<td></td>
<td>GLO 8</td>
<td>;If keyup timer zero, then</td>
</tr>
<tr>
<td>0415</td>
<td>32</td>
<td>21</td>
<td>BZ STDN</td>
<td>;Key was down before, so jump.</td>
</tr>
<tr>
<td>0417</td>
<td>FF</td>
<td>40</td>
<td>SMI 40</td>
<td>;If timer = 40, timer has reached limit, so jump.</td>
</tr>
<tr>
<td>0419</td>
<td>32</td>
<td>21</td>
<td>BZ STDN</td>
<td></td>
</tr>
<tr>
<td>041B</td>
<td>D4</td>
<td>0540</td>
<td>SUB UBIN</td>
<td>;Else sub to bin key up time</td>
</tr>
<tr>
<td>041E</td>
<td>C1</td>
<td>0568</td>
<td>L3Q AUT?</td>
<td>;If Q was set by BIN, then a histogram bin was full, so jump to AUT? for action.</td>
</tr>
<tr>
<td>0421</td>
<td>F8</td>
<td>00</td>
<td>STDN LDI 00</td>
<td>;Reset Key up timer.</td>
</tr>
<tr>
<td>0423</td>
<td>AC</td>
<td></td>
<td>PLO C</td>
<td></td>
</tr>
<tr>
<td>0424</td>
<td>89</td>
<td></td>
<td>GLO 9</td>
<td>;Get Key down timer.</td>
</tr>
<tr>
<td>0425</td>
<td>FF</td>
<td>40</td>
<td>SMI 40</td>
<td>;subtract 40, zero if full.</td>
</tr>
<tr>
<td>0428</td>
<td>32</td>
<td>00</td>
<td>BZ TIME</td>
<td>;so jump back to timer.</td>
</tr>
<tr>
<td>042A</td>
<td>19</td>
<td></td>
<td>INC 9</td>
<td>;Otherwise increment Key</td>
</tr>
<tr>
<td>042A</td>
<td>30</td>
<td>00</td>
<td>BR TIME</td>
<td>;down counter and go to timer</td>
</tr>
<tr>
<td>042C</td>
<td>8C</td>
<td></td>
<td>GLO C</td>
<td>;Here if key up, get timer.</td>
</tr>
<tr>
<td>042D</td>
<td>3A</td>
<td>58</td>
<td>BNZ STUP</td>
<td>;Jump if 0, was up before.</td>
</tr>
<tr>
<td>042F</td>
<td>9E</td>
<td></td>
<td>GHI E</td>
<td>;Else get mark threshold.</td>
</tr>
<tr>
<td>0430</td>
<td>57</td>
<td></td>
<td>STR 7</td>
<td>;Put it in free location.</td>
</tr>
<tr>
<td>0431</td>
<td>89</td>
<td></td>
<td>GLO 9</td>
<td>;Get Key down timer, find</td>
</tr>
<tr>
<td>0432</td>
<td>F7</td>
<td></td>
<td>SM</td>
<td>;Key down - mark threshold.</td>
</tr>
<tr>
<td>0433</td>
<td>33</td>
<td>3D</td>
<td>BCE DASH</td>
<td>;Jump if &gt;= 0, it was a dash.</td>
</tr>
<tr>
<td>0435</td>
<td>99</td>
<td></td>
<td>GHI 9</td>
<td>;Get Morse shift register.</td>
</tr>
<tr>
<td>0436</td>
<td>FE</td>
<td></td>
<td>SHL</td>
<td>;Shift left, 0=dot into lsb.</td>
</tr>
<tr>
<td>0437</td>
<td>B9</td>
<td></td>
<td>PHI 9</td>
<td>;Put it back.</td>
</tr>
<tr>
<td>0438</td>
<td>89</td>
<td></td>
<td>GLO 9</td>
<td>;Get timer AGAIN!!!</td>
</tr>
<tr>
<td>0439</td>
<td>FF</td>
<td>08</td>
<td>SMI 08</td>
<td>;Subst. 08, 0 if perfect dot.</td>
</tr>
<tr>
<td>043B</td>
<td>30</td>
<td>45</td>
<td>BR SPD?</td>
<td>;Jump ahead to check speed.</td>
</tr>
<tr>
<td>043D</td>
<td>99</td>
<td></td>
<td>DASH GHI 9</td>
<td>;Get shift register.</td>
</tr>
<tr>
<td>043E</td>
<td>FE</td>
<td></td>
<td>SHL</td>
<td>;Shift left, 0 into lsb.</td>
</tr>
<tr>
<td>043F</td>
<td>FC</td>
<td>01</td>
<td>ADI 01</td>
<td>;Add 01, 1=dash into lsb.</td>
</tr>
<tr>
<td>0441</td>
<td>B9</td>
<td></td>
<td>PHI 9</td>
<td>;Put it back.</td>
</tr>
<tr>
<td>0442</td>
<td>89</td>
<td></td>
<td>GLO 9</td>
<td>;Get timer again.</td>
</tr>
<tr>
<td>0443</td>
<td>FF</td>
<td>18</td>
<td>SMI 18</td>
<td>;Subst. 18, 0 if perfect dash.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>;At this point, the D register is zero if the dot or dash was of a perfect length. DF=1 if the speed is too fast ;(dot &gt; 08 or dash &gt; 18), and DF=0 if too slow.</td>
</tr>
<tr>
<td>0445</td>
<td>32</td>
<td>4B</td>
<td>SPD? BZ UP1</td>
<td>;Jump if D=0, no change.</td>
</tr>
<tr>
<td>0447</td>
<td>1D</td>
<td></td>
<td>INC D</td>
<td>;Inc. (slower) timing constant</td>
</tr>
<tr>
<td>0448</td>
<td>CF</td>
<td></td>
<td>LSDF</td>
<td>;Skip 2 if DF=1, speed was</td>
</tr>
<tr>
<td>0449</td>
<td>2D</td>
<td></td>
<td>DEC D</td>
<td>;fast, else dec. by 2, effec-</td>
</tr>
<tr>
<td>044A</td>
<td>2D</td>
<td></td>
<td>DEC D</td>
<td>;tively by 1, raise speed.</td>
</tr>
<tr>
<td>044B</td>
<td>D4</td>
<td>0545</td>
<td>SUB DBIN</td>
<td>;Sub to bin key down.</td>
</tr>
<tr>
<td>044E</td>
<td>39</td>
<td>54</td>
<td>BNQ UP2</td>
<td>;Jmp if Q=0, no bins full,</td>
</tr>
<tr>
<td>0450</td>
<td>7A</td>
<td></td>
<td>REQ</td>
<td>;else reset Q,</td>
</tr>
<tr>
<td>0451</td>
<td>C0</td>
<td>0568</td>
<td>LBR AUT?</td>
<td>;and jump for action.</td>
</tr>
<tr>
<td>Add.</td>
<td>Code</td>
<td>Label</td>
<td>Mnemonic</td>
<td>Comments</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>0454</td>
<td>9C</td>
<td>UP2</td>
<td>GHI C</td>
<td>Get counter for number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>key downs in a letter incre-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>mental and restore it.</td>
</tr>
<tr>
<td>0455</td>
<td>FC 01</td>
<td>ADI 01</td>
<td>PHC</td>
<td>Reset key down timer.</td>
</tr>
<tr>
<td>0456</td>
<td>BC</td>
<td>PHI C</td>
<td></td>
<td>Get key up timer,</td>
</tr>
<tr>
<td>0457</td>
<td>F8 00</td>
<td>STUP</td>
<td>LDI 00</td>
<td>subtr. 40, 0 if timer full,</td>
</tr>
<tr>
<td>0458</td>
<td>A9</td>
<td>PLO</td>
<td>9</td>
<td>jump if so.</td>
</tr>
<tr>
<td>0459</td>
<td>BC</td>
<td>GLO C</td>
<td></td>
<td>If RC.0=0, there are no</td>
</tr>
<tr>
<td>0460</td>
<td>FF 40</td>
<td>SMI 40</td>
<td></td>
<td>elements in letter, so jump,</td>
</tr>
<tr>
<td>0461</td>
<td>32 79</td>
<td>BZ UP3</td>
<td></td>
<td>Get space threshold, and put</td>
</tr>
<tr>
<td>0462</td>
<td>8E</td>
<td>STR 7</td>
<td>GLO E</td>
<td>it in a free location.</td>
</tr>
<tr>
<td>0463</td>
<td>57</td>
<td>SM</td>
<td>GLO C</td>
<td>Get key up timer and find</td>
</tr>
<tr>
<td>0464</td>
<td>BC</td>
<td>GLO C</td>
<td></td>
<td>key up - space threshold.</td>
</tr>
<tr>
<td>0465</td>
<td>SE</td>
<td>GLO C</td>
<td></td>
<td>If 0, now output letter.</td>
</tr>
<tr>
<td>0466</td>
<td>UP4</td>
<td>GLO C</td>
<td></td>
<td>Find inter-character space:</td>
</tr>
<tr>
<td>0467</td>
<td>32 80</td>
<td>BZ LOUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0468</td>
<td>8E</td>
<td>SHR</td>
<td></td>
<td>ICS = 4*RE.0 - RE.0/2</td>
</tr>
<tr>
<td>0469</td>
<td>57</td>
<td>STR 7</td>
<td>GLO E</td>
<td>= 3.5*RE.0</td>
</tr>
<tr>
<td>0470</td>
<td>FE</td>
<td>SHL</td>
<td>GLO C</td>
<td>For perfect morse it should</td>
</tr>
<tr>
<td>0471</td>
<td>FE</td>
<td>SHL</td>
<td></td>
<td>be 3.0, but hand sent is</td>
</tr>
<tr>
<td>0472</td>
<td>F7</td>
<td>SM</td>
<td>GLO C</td>
<td>better with this ratio.</td>
</tr>
<tr>
<td>0473</td>
<td>3A 78</td>
<td>BNZ UP5</td>
<td></td>
<td>Put ICS in free location.</td>
</tr>
<tr>
<td>0474</td>
<td>F8 20</td>
<td>LDI 20</td>
<td></td>
<td>Get key up timer and find</td>
</tr>
<tr>
<td>0475</td>
<td>DA</td>
<td>SEP A</td>
<td></td>
<td>Key up - ICS</td>
</tr>
<tr>
<td>0476</td>
<td>1C</td>
<td>INC A</td>
<td></td>
<td>Jump if not zero, no space.</td>
</tr>
<tr>
<td>0477</td>
<td>3F 00</td>
<td>UP3</td>
<td>LDI 20</td>
<td>Load ASCII &quot;space&quot;,</td>
</tr>
<tr>
<td>0478</td>
<td>30 B8</td>
<td>BR KCHK</td>
<td></td>
<td>Output to serial port 1</td>
</tr>
<tr>
<td>0479</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td>Increment key up timer.</td>
</tr>
<tr>
<td>0480</td>
<td>00</td>
<td>IDL</td>
<td></td>
<td>Jump to timer if no key-</td>
</tr>
<tr>
<td>0481</td>
<td>99</td>
<td>LOUT</td>
<td></td>
<td>board input, else go and</td>
</tr>
<tr>
<td>0482</td>
<td>9C</td>
<td>CHI 9</td>
<td></td>
<td>see what it is.</td>
</tr>
<tr>
<td>0483</td>
<td>9C</td>
<td>CHI C</td>
<td></td>
<td>Following section outputs</td>
</tr>
<tr>
<td>0484</td>
<td>9C</td>
<td>SMI 06</td>
<td></td>
<td>a letter.</td>
</tr>
<tr>
<td>0485</td>
<td>FF 06</td>
<td>BNZ N06</td>
<td></td>
<td>Get Morse shift register,</td>
</tr>
<tr>
<td>0486</td>
<td>27</td>
<td>DEC 7</td>
<td></td>
<td>put it in a free location,</td>
</tr>
<tr>
<td>0487</td>
<td>9A</td>
<td>CHI C</td>
<td></td>
<td>Output it to 8 leds for a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SMI 06</td>
<td></td>
<td>visual check of dots/dashes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BNZ N06</td>
<td></td>
<td>Check if 6 key downs in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>letter, jump if not for</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>standard decoding.</td>
</tr>
</tbody>
</table>
MORSE PROGRAM

;The next segment of code is for Morse characters with
;exactly six elements as some require different decoding.

<table>
<thead>
<tr>
<th>Add.</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0489</td>
<td>99</td>
<td>GHI 9</td>
<td></td>
<td>Get Morse shift register and</td>
</tr>
<tr>
<td>048A</td>
<td>FF 4C</td>
<td></td>
<td>SMI 4C</td>
<td>subtract Morse &quot;?&quot;,</td>
</tr>
<tr>
<td>048C</td>
<td>3A 92</td>
<td></td>
<td>BNZ NO?</td>
<td>Jump if not that,</td>
</tr>
<tr>
<td>048E</td>
<td>F8 3F</td>
<td></td>
<td>LDI 3F</td>
<td>else load ASCII &quot;?&quot; and</td>
</tr>
<tr>
<td>0490</td>
<td>30 A8</td>
<td></td>
<td>BR MOUT</td>
<td>jump to output it,</td>
</tr>
<tr>
<td>0492</td>
<td>FF 2C</td>
<td></td>
<td>NO?</td>
<td>Subtract further offset,</td>
</tr>
<tr>
<td>0494</td>
<td>3A A0</td>
<td></td>
<td>BNZ OKLET</td>
<td>jump if not 0,</td>
</tr>
<tr>
<td>0496</td>
<td>F8 3A</td>
<td></td>
<td>LDI 3A</td>
<td>else load ASCII &quot;:&quot; and</td>
</tr>
<tr>
<td>0498</td>
<td>30 A8</td>
<td></td>
<td>BR MOUT</td>
<td>jump to output it.</td>
</tr>
</tbody>
</table>

;The remaining 6 element characters are handled normally.
;At this point the D register holds # of key downs - six.

<table>
<thead>
<tr>
<th>Add.</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>049A</td>
<td>3B A0</td>
<td>NO6</td>
<td>BL OKLET</td>
<td>If less than six, ok, but if</td>
</tr>
<tr>
<td>049C</td>
<td>F8 2A</td>
<td></td>
<td>LDI 2A</td>
<td>not load ASCII &quot;*&quot; for over-</td>
</tr>
<tr>
<td>049E</td>
<td>30 A8</td>
<td></td>
<td>BR MOUT</td>
<td>run and jump to output it.</td>
</tr>
<tr>
<td>04A0</td>
<td>99</td>
<td>OKLET</td>
<td>GHI 9</td>
<td>Get Morse shift register,</td>
</tr>
<tr>
<td>04A1</td>
<td>FA 3F</td>
<td></td>
<td>ANI 3F</td>
<td>mask in lower 6 bits, put</td>
</tr>
<tr>
<td>04A3</td>
<td>AF</td>
<td></td>
<td>PLO F</td>
<td>it in RF.0 as low address.</td>
</tr>
<tr>
<td>04A4</td>
<td>F8 05</td>
<td></td>
<td>LDI 05</td>
<td>Put high address in RF.1,</td>
</tr>
<tr>
<td>04A6</td>
<td>BF</td>
<td></td>
<td>PHI F</td>
<td>point to Morse/ASCII table.</td>
</tr>
<tr>
<td>04A7</td>
<td>0F</td>
<td></td>
<td>LDN F</td>
<td>Get ASCII</td>
</tr>
<tr>
<td>04A8</td>
<td>DA</td>
<td>MOUT</td>
<td>SEP A</td>
<td>and output it.</td>
</tr>
<tr>
<td>04A9</td>
<td>1C</td>
<td></td>
<td>INC C</td>
<td>Increment key up timer.</td>
</tr>
<tr>
<td>04AA</td>
<td>27</td>
<td></td>
<td>DEC 7</td>
<td>Point to free memory (07EF).</td>
</tr>
<tr>
<td>04AB</td>
<td>8C</td>
<td></td>
<td>GLO C</td>
<td>Get key up timer,</td>
</tr>
<tr>
<td>04AC</td>
<td>57</td>
<td></td>
<td>STR 7</td>
<td>save it in memory, and point</td>
</tr>
<tr>
<td>04AD</td>
<td>60</td>
<td></td>
<td>IRX</td>
<td>back to I/O location (07F0).</td>
</tr>
<tr>
<td>04AE</td>
<td>C0</td>
<td>0600</td>
<td>LBR THRM</td>
<td>Jump to find new thresholds.</td>
</tr>
<tr>
<td>04B1</td>
<td>00</td>
<td></td>
<td>IDL</td>
<td></td>
</tr>
<tr>
<td>04B2</td>
<td>00</td>
<td></td>
<td>IDL</td>
<td></td>
</tr>
<tr>
<td>04B3</td>
<td>00</td>
<td></td>
<td>IDL</td>
<td></td>
</tr>
<tr>
<td>04B4</td>
<td>00</td>
<td></td>
<td>IDL</td>
<td></td>
</tr>
<tr>
<td>04B5</td>
<td>00</td>
<td></td>
<td>IDL</td>
<td></td>
</tr>
<tr>
<td>04B6</td>
<td>00</td>
<td></td>
<td>IDL</td>
<td></td>
</tr>
<tr>
<td>04B7</td>
<td>00</td>
<td></td>
<td>IDL</td>
<td></td>
</tr>
</tbody>
</table>
**MORSE PROGRAM**

The next segment of code checks for input from the ASCII keyboard and acts accordingly.

<table>
<thead>
<tr>
<th>Add.</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>04B8</td>
<td>3F B8</td>
<td>KCHK</td>
<td>BN4 KCHK</td>
<td>Loop until keyboard input.</td>
</tr>
<tr>
<td>04BA</td>
<td>6B</td>
<td></td>
<td></td>
<td>Input keyboard into D, M(RX)</td>
</tr>
<tr>
<td>04BB</td>
<td>FF 2F</td>
<td></td>
<td></td>
<td>Subtract ASCII &quot;/&quot;.</td>
</tr>
<tr>
<td>04BD</td>
<td>C2 03DD</td>
<td>LBZ</td>
<td>RSTRT</td>
<td>Complete restart if &quot;/&quot;.</td>
</tr>
<tr>
<td>04C0</td>
<td>FF 19</td>
<td></td>
<td></td>
<td>Subtract again, 0 if &quot;H&quot;.</td>
</tr>
<tr>
<td>04C2</td>
<td>C2 0250</td>
<td>LBZ</td>
<td>HIST</td>
<td>Output histogram if &quot;H&quot;.</td>
</tr>
<tr>
<td>04C5</td>
<td>FF 03</td>
<td></td>
<td></td>
<td>Subtract again, 0 if &quot;K&quot;.</td>
</tr>
<tr>
<td>04C7</td>
<td>3A D7</td>
<td></td>
<td></td>
<td>Jump if not, else point</td>
</tr>
<tr>
<td>04C9</td>
<td>FB 10</td>
<td></td>
<td></td>
<td>Register A to output</td>
</tr>
<tr>
<td>04CB</td>
<td>AA</td>
<td></td>
<td></td>
<td>Routine for serial port 2.</td>
</tr>
<tr>
<td>04CC</td>
<td>F8 1B</td>
<td></td>
<td></td>
<td>Output &quot;ESC&quot; &amp; &quot;ETB&quot; to</td>
</tr>
<tr>
<td>04CE</td>
<td>DA</td>
<td></td>
<td></td>
<td>Serial port 2. This sends</td>
</tr>
<tr>
<td>04CF</td>
<td>F8 17</td>
<td></td>
<td></td>
<td>a copy command to the</td>
</tr>
<tr>
<td>04D1</td>
<td>DA</td>
<td></td>
<td></td>
<td>Tektronix Graphics Terminal.</td>
</tr>
<tr>
<td>04D2</td>
<td>F8 01</td>
<td></td>
<td></td>
<td>Point register A back to the</td>
</tr>
<tr>
<td>04D4</td>
<td>AA</td>
<td></td>
<td></td>
<td>Routine for serial port 1.</td>
</tr>
<tr>
<td>04D5</td>
<td>30 B8</td>
<td></td>
<td></td>
<td>Back for more input.</td>
</tr>
<tr>
<td>04D7</td>
<td>FF 02</td>
<td>NOKO</td>
<td></td>
<td>Subtract again, 0 if &quot;M&quot;.</td>
</tr>
<tr>
<td>04D9</td>
<td>C2 03F6</td>
<td>LBZ</td>
<td>RSTR1</td>
<td>If so partial restart.</td>
</tr>
<tr>
<td>04DC</td>
<td>FF 01</td>
<td></td>
<td></td>
<td>Subtract again, 0 if &quot;N&quot;.</td>
</tr>
<tr>
<td>04DE</td>
<td>32 F6</td>
<td></td>
<td></td>
<td>If so, jump to cancel auto.</td>
</tr>
<tr>
<td>04E0</td>
<td>FF 0A</td>
<td></td>
<td></td>
<td>Subtract again, 0 if &quot;X&quot;.</td>
</tr>
<tr>
<td>04E2</td>
<td>7A</td>
<td></td>
<td></td>
<td>Reset Q (will clear hist.)</td>
</tr>
<tr>
<td>04E3</td>
<td>32 EA</td>
<td></td>
<td></td>
<td>If &quot;X&quot;, go clear histograms.</td>
</tr>
<tr>
<td>04E5</td>
<td>FF 01</td>
<td></td>
<td></td>
<td>Subtract again, 0 if &quot;Y&quot;.</td>
</tr>
<tr>
<td>04E7</td>
<td>3A F1</td>
<td></td>
<td></td>
<td>Jump if not, else</td>
</tr>
<tr>
<td>04E9</td>
<td>7B</td>
<td></td>
<td></td>
<td>Set Q (will histogram/2)</td>
</tr>
<tr>
<td>04EA</td>
<td>D4 03C0 COMAS</td>
<td></td>
<td></td>
<td>and divide hist. data by 2.</td>
</tr>
<tr>
<td>04ED</td>
<td>7A</td>
<td></td>
<td></td>
<td>Reset Q (X &amp; Y return here)</td>
</tr>
<tr>
<td>04EE</td>
<td>C0 03F6</td>
<td>BR</td>
<td>RSTR1</td>
<td>and do partial restart.</td>
</tr>
<tr>
<td>04F1</td>
<td>FF 01</td>
<td>Z?</td>
<td></td>
<td>Subtract AGAIN!, 0 if &quot;Z&quot;.</td>
</tr>
<tr>
<td>04F3</td>
<td>CA 0000</td>
<td></td>
<td></td>
<td>If not, invalid, to monitor.</td>
</tr>
<tr>
<td>04F6</td>
<td>27</td>
<td>SAUT</td>
<td>DEC 7</td>
<td>Here if &quot;Z&quot; or &quot;N&quot;, point to</td>
</tr>
<tr>
<td>04F7</td>
<td>27</td>
<td></td>
<td></td>
<td>Location for auto command,</td>
</tr>
<tr>
<td>04F8</td>
<td>6B</td>
<td></td>
<td></td>
<td>Input ASKEY (07EE) and put input there.</td>
</tr>
<tr>
<td>04F9</td>
<td>60</td>
<td>IRX</td>
<td></td>
<td>&quot;Z&quot; for auto, &quot;N&quot; nonauto.</td>
</tr>
<tr>
<td>04FA</td>
<td>60</td>
<td>IRX</td>
<td></td>
<td>Point back to I/O location.</td>
</tr>
<tr>
<td>04FB</td>
<td>30 00</td>
<td>BR</td>
<td>TIME</td>
<td>Go back to timing loop.</td>
</tr>
<tr>
<td>04FD</td>
<td>00</td>
<td></td>
<td></td>
<td>IDL</td>
</tr>
<tr>
<td>04FE</td>
<td>00</td>
<td></td>
<td></td>
<td>IDL</td>
</tr>
<tr>
<td>04FF</td>
<td>00</td>
<td></td>
<td></td>
<td>IDL</td>
</tr>
</tbody>
</table>
MORSE PROGRAM

;This is the lookup table to convert Morse to ASCII.
;A register is preset to 0000 0001 and the elements are
;shifted into the lsb, 0 for dot and 1 for dash. The
;leading 1 shows the beginning of the character. The letter
;"L" is ".-" and would appear as 0001 0100. The upper two
;bits are set to 00. This forms the lower eight bits of the
;address, and 05 is the offset for the high eight bits.
;This scheme is memory efficient, but causes characters
;with six elements to show up anywhere in the table.
;This is only a problem for "?" and ";", so these are dealt
;with in software. "@" is used for an invalid character,
;and "<" for the Morse error "-----".

<table>
<thead>
<tr>
<th>Add.</th>
<th>Data</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>0500</td>
<td>3C</td>
<td>&lt;</td>
</tr>
<tr>
<td>0501</td>
<td>40</td>
<td>@</td>
</tr>
<tr>
<td>0502</td>
<td>45</td>
<td>E</td>
</tr>
<tr>
<td>0503</td>
<td>54</td>
<td>T</td>
</tr>
<tr>
<td>0504</td>
<td>49</td>
<td>I</td>
</tr>
<tr>
<td>0505</td>
<td>41</td>
<td>A</td>
</tr>
<tr>
<td>0506</td>
<td>4E</td>
<td>N</td>
</tr>
<tr>
<td>0507</td>
<td>4D</td>
<td>M</td>
</tr>
<tr>
<td>0508</td>
<td>53</td>
<td>S</td>
</tr>
<tr>
<td>0509</td>
<td>55</td>
<td>U</td>
</tr>
<tr>
<td>050A</td>
<td>52</td>
<td>R</td>
</tr>
<tr>
<td>050B</td>
<td>57</td>
<td>W</td>
</tr>
<tr>
<td>050C</td>
<td>44</td>
<td>D</td>
</tr>
<tr>
<td>050D</td>
<td>4B</td>
<td>K</td>
</tr>
<tr>
<td>050E</td>
<td>47</td>
<td>G</td>
</tr>
<tr>
<td>050F</td>
<td>4F</td>
<td>O</td>
</tr>
<tr>
<td>0510</td>
<td>48</td>
<td>H</td>
</tr>
<tr>
<td>0511</td>
<td>56</td>
<td>V</td>
</tr>
<tr>
<td>0512</td>
<td>46</td>
<td>F</td>
</tr>
<tr>
<td>0513</td>
<td>40</td>
<td>@</td>
</tr>
<tr>
<td>0514</td>
<td>4C</td>
<td>L</td>
</tr>
<tr>
<td>0515</td>
<td>2E</td>
<td>.</td>
</tr>
<tr>
<td>0516</td>
<td>50</td>
<td>P</td>
</tr>
<tr>
<td>0517</td>
<td>4A</td>
<td>J</td>
</tr>
<tr>
<td>0518</td>
<td>42</td>
<td>B</td>
</tr>
<tr>
<td>0519</td>
<td>58</td>
<td>X</td>
</tr>
<tr>
<td>0520</td>
<td>35</td>
<td>5</td>
</tr>
<tr>
<td>0521</td>
<td>34</td>
<td>4</td>
</tr>
<tr>
<td>0522</td>
<td>40</td>
<td>@</td>
</tr>
<tr>
<td>0523</td>
<td>33</td>
<td>3</td>
</tr>
<tr>
<td>0524</td>
<td>40</td>
<td>@</td>
</tr>
<tr>
<td>0525</td>
<td>40</td>
<td>@</td>
</tr>
<tr>
<td>0526</td>
<td>33</td>
<td>3</td>
</tr>
<tr>
<td>0527</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>0528</td>
<td>40</td>
<td>@</td>
</tr>
<tr>
<td>0529</td>
<td>40</td>
<td>@</td>
</tr>
<tr>
<td>052A</td>
<td>3B</td>
<td>;</td>
</tr>
<tr>
<td>052B</td>
<td>40</td>
<td>@</td>
</tr>
<tr>
<td>052C</td>
<td>40</td>
<td>@</td>
</tr>
<tr>
<td>052D</td>
<td>5B</td>
<td>[</td>
</tr>
<tr>
<td>052E</td>
<td>40</td>
<td>@</td>
</tr>
<tr>
<td>052F</td>
<td>31</td>
<td>1</td>
</tr>
<tr>
<td>0530</td>
<td>36</td>
<td>6</td>
</tr>
<tr>
<td>0531</td>
<td>5F</td>
<td>-</td>
</tr>
<tr>
<td>0532</td>
<td>2F</td>
<td>/</td>
</tr>
<tr>
<td>0533</td>
<td>2C</td>
<td>,</td>
</tr>
<tr>
<td>0534</td>
<td>40</td>
<td>@</td>
</tr>
<tr>
<td>0535</td>
<td>40</td>
<td>@</td>
</tr>
<tr>
<td>0536</td>
<td>40</td>
<td>@</td>
</tr>
<tr>
<td>0537</td>
<td>40</td>
<td>@</td>
</tr>
<tr>
<td>0538</td>
<td>37</td>
<td>7</td>
</tr>
<tr>
<td>0539</td>
<td>40</td>
<td>@</td>
</tr>
<tr>
<td>053A</td>
<td>40</td>
<td>@</td>
</tr>
<tr>
<td>053B</td>
<td>40</td>
<td>@</td>
</tr>
<tr>
<td>053C</td>
<td>38</td>
<td>8</td>
</tr>
<tr>
<td>053D</td>
<td>40</td>
<td>@</td>
</tr>
<tr>
<td>053E</td>
<td>39</td>
<td>9</td>
</tr>
<tr>
<td>053F</td>
<td>30</td>
<td>0</td>
</tr>
</tbody>
</table>
MORSE PROGRAM

; This is the subroutine to enter the mark and space times; into their respective histograms. The raw space data is; stored from 0580 to 05BF, the raw mark data from 05C0; to 05FF.

<table>
<thead>
<tr>
<th>Addr.</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0540</td>
<td>BC</td>
<td>UBIN</td>
<td>GLO C</td>
<td>Enter here to bin space, get</td>
</tr>
<tr>
<td>0541</td>
<td>FC 80</td>
<td>ADI</td>
<td>SMI 80</td>
<td>Key up timer, add offset,</td>
</tr>
<tr>
<td>0543</td>
<td>30 4D</td>
<td>BR SP</td>
<td></td>
<td>and jump ahead.</td>
</tr>
<tr>
<td>0545</td>
<td>89</td>
<td>DBIN</td>
<td>GLO 9</td>
<td>Enter here for mark, get</td>
</tr>
<tr>
<td>0546</td>
<td>FF 40</td>
<td>SMI</td>
<td>40</td>
<td>Timer, subtract 40, jump if</td>
</tr>
<tr>
<td>0548</td>
<td>33 5F</td>
<td>BCE</td>
<td>BIGM</td>
<td>&gt;=0, therefore overflow.</td>
</tr>
<tr>
<td>054A</td>
<td>89</td>
<td>GLO</td>
<td>9</td>
<td>If ok, get timer again, add</td>
</tr>
<tr>
<td>054B</td>
<td>FC C0</td>
<td>ADI</td>
<td>C0</td>
<td>Offset.</td>
</tr>
<tr>
<td>054D</td>
<td>AF</td>
<td>SP</td>
<td>PLO F</td>
<td>For mark or space, put data</td>
</tr>
<tr>
<td>054E</td>
<td>FB 05</td>
<td>LDI</td>
<td>05</td>
<td>Plus offset into RF.0, and</td>
</tr>
<tr>
<td>0550</td>
<td>BF</td>
<td>PHI</td>
<td>F</td>
<td>High address into RF.1.</td>
</tr>
</tbody>
</table>

; The value from the timer, e.g., 2C, is used to point to the; 2Cth bin of the appropriate histogram. The number in the; 2Cth bin indicates the number of times the particular; timer reached exactly 2C before the key changed.

<table>
<thead>
<tr>
<th>Addr.</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0551</td>
<td>0F</td>
<td>LDN</td>
<td>F</td>
<td>Get the data from that bin.</td>
</tr>
<tr>
<td>0552</td>
<td>FF 3F</td>
<td>SMI</td>
<td>3F</td>
<td>Subt. 3F (63), 0 if full.</td>
</tr>
<tr>
<td>0554</td>
<td>32 5C</td>
<td>SEQ</td>
<td></td>
<td>Set Q in case it's full.</td>
</tr>
<tr>
<td>0555</td>
<td>7A</td>
<td>BZ</td>
<td>BIND</td>
<td>If full, jump out with Q=1.</td>
</tr>
<tr>
<td>0557</td>
<td>0F</td>
<td>LDN</td>
<td>F</td>
<td>And get data again from bin.</td>
</tr>
<tr>
<td>0559</td>
<td>FC 01</td>
<td>ADI</td>
<td>01</td>
<td>Increment the data</td>
</tr>
<tr>
<td>055B</td>
<td>5F</td>
<td>STR</td>
<td>F</td>
<td>And restore it.</td>
</tr>
<tr>
<td>055C</td>
<td>C0 0760</td>
<td>BIND</td>
<td>BR SMTH</td>
<td>Done, jmp to smooth routine.</td>
</tr>
</tbody>
</table>

; Program ends up here if key down timer overflowed. This; usually means the overall speed is far too high.

<table>
<thead>
<tr>
<th>Addr.</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>055F</td>
<td>8D</td>
<td>BIGM</td>
<td>GLO D</td>
<td>The timing constant is</td>
</tr>
<tr>
<td>0560</td>
<td>FE</td>
<td>SHL</td>
<td></td>
<td>Stored in RD, and this is</td>
</tr>
<tr>
<td>0561</td>
<td>AD</td>
<td>PLO</td>
<td>D</td>
<td>Shifted left by one in a 16</td>
</tr>
<tr>
<td>0562</td>
<td>9D</td>
<td>CHI</td>
<td>D</td>
<td>Bit shift. This multiplies</td>
</tr>
<tr>
<td>0563</td>
<td>7E</td>
<td>SHLC</td>
<td></td>
<td>The constant by 2 and halves</td>
</tr>
<tr>
<td>0564</td>
<td>BD</td>
<td>PHI</td>
<td>D</td>
<td>The speed of the timer.</td>
</tr>
<tr>
<td>0565</td>
<td>C0 03EF</td>
<td>LBR</td>
<td>RSTR2</td>
<td>Do a partial restart with</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the new slower speed and zeroed histograms.</td>
</tr>
</tbody>
</table>
MORSE PROGRAM

;The binning subroutine sets the Q flag if a bin is full.
;On return to the key up or key down routines the flag
;is sampled. If set, control passes to this routine which
;checks the auto histogram flag stored in 07EE. This
;is not a subroutine as it can return to different places.
;If entered from the key down section, Q is still set, but
;the key up section resets Q first. Q tells this section
;where to return if no histogram was required.

Add. Code Label Mnemonic Comments
0568 27 AUT? DEC 7 ;Point to location which
0569 27 DEC 7 ;holds auto flag.
056A 72 LDXA ;Get it and point back to
056B 60 IRX ;free location (07F0).
056C FF 5A SMI 5A ;Subtract 5A, 0 if "Z".
056E C2 0250 LBZ HIST ;If so, jump for histogram.
0571 C9 0454 LBNQ UP2 ;If Q=0, back to key up,
0574 7A REQ ;else reset Q
0575 CO 0421 LBR STDN ;and back to key down.
0578 00 IDL ;IDL's to 057F inc.

;0580 to 05BF contains raw histogram data for spaces.
;05C0 to 05FF contains raw histogram data for marks.

;This is the subroutine to clear (Q=0) or divide by two
;(Q=1) the raw histogram data.

Add. Code Label Mnemonic Comments
03C0 F8 05 MASG LDI 05 ;Put high address pointer
03C2 BF PHI F ;into RF.1.
03C3 F8 80 LDI 80 ;Low address is start of raw
03C5 AF PLO F ;space data, marks follow.
03C6 8F MAS? GLO F ;If RF.0=00, then done.
03C7 C6 LSNZ ;Skip two if not done,
03C8 D5 RETURN ;else return.
03C9 00 IDL ;
03CA C5 LSNQ ;Skip two if Q=0 (clear),
03CB 0F LDN F ;else get data from bin and
03CC F6 SHR ;shift right to halve it.
03CD CD LSQ ;Skip two if Q=1 (halve),
03CE F8 00 LDI 00 ;else load 00 to clear.
03D0 5F STR F ;Put new data into bin
03D1 1F INC F ;and increment pointer.
03D2 30 C6 BR MAS? ;Loop back for more.
03D4 00 IDL ;IDL's to 03DC inc.
MORSE PROGRAM

;This is the smoothing subroutine. It is a continuation of
;the binning subroutine which returns from here. The raw
;histogram data from 0580-05BF, 05C0-05FF is smoothed by
;a simple algorithm and stored from 0680-06BF, 06C0-06FF.
;The Q flag is still set if a bin was full.

Add.  Code  Label  Mnemonic  Comments
0760  F8  05  SMTH  LDI  05  ;Load 0580 into RF, point to
0762  BF  PHI  F  ;raw data, 0680 into R0,
0763  F8  06  LDI  06  ;space for smoothed data.
0765  B0  PHI  0
0766  F8  80  LDI  80
0768  A0  PLO  0
0769  AF  PLO  F
076A  E0  SEX  0  ;RO used for math, make it RX
076B  0F  SMTH1  LDN  F  ;Get raw data(N)
076C  FE  SHL  ;multiply by 2,
076D  50  STR  0  ;store it in smooth space.
076E  2F  DEC  F  ;Point to raw data(N-1).
076F  8F  GLO  F  ;If RF=057F, out of data
0770  FF  7F  SMI  7F  ;storage area,
0772  32  7B  BZ  BDAT  ;so jump ahead.
0774  FF  40  SMI  40  ;If RF=05BF, marks intruding
0776  32  7B  BZ  BDAT  ;into spaces, so jump ahead.
0778  0F  LDN  F  ;Get raw data(N-1) and add it
0779  F4  ADD  ;to 2*raw data(N).
077A  50  STR  0  ;Put result in smoothed area.
077B  1F  BDAT  INC  F  ;Point to raw data(N+1).
077C  1F  INC  F
077D  8F  GLO  F  ;If RF=0600, out of data,
077E  32  87  BZ  BDAT2  ;so jump ahead.
0780  FF  80  LDI  80  ;If RF=0580, spaces intruding
0782  32  87  BZ  BDAT2  ;into marks, so jump ahead.
0784  0F  LDN  F  ;Get raw data(N+1) and add it
0785  F4  ADD  ;to value from line 0779
0786  50  STR  0  ;Put result in smoothed area.
0787  F0  BDAT2  LDX  ;Get result so far,
0788  FC  02  ADI  02  ;add two for rounding.
078A  F6  SHR  ;Divide by four
078B  F6  SHR
078C  50  STR  0  ;and store result.

;The data was smoothed with the following formula:
;smooth(n) = (2*raw(n) + raw(n-1) + raw(n+1) + 2)/4
### MORSE PROGRAM

;This is the continuation of the smoothing routine.

<table>
<thead>
<tr>
<th>Add. Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>078D 60</td>
<td></td>
<td>IRX</td>
<td>Point to next smooth space.</td>
</tr>
<tr>
<td>078E 80</td>
<td></td>
<td>CLO 0</td>
<td>If R0 not 0700, not done.</td>
</tr>
<tr>
<td>078F 3A 6B</td>
<td></td>
<td>BNZ SMTH1</td>
<td>Jump back for more.</td>
</tr>
<tr>
<td>0791 E7</td>
<td></td>
<td>SEX 7</td>
<td>Else restore R7 as RX.</td>
</tr>
<tr>
<td>0792 CD</td>
<td></td>
<td>LSQ</td>
<td>If Q=0, no bins full.</td>
</tr>
<tr>
<td>0793 D5</td>
<td></td>
<td>RETURN</td>
<td>Return, otherwise skip 2.</td>
</tr>
<tr>
<td>0794 00</td>
<td></td>
<td>IDL</td>
<td></td>
</tr>
<tr>
<td>0795 D4 03C0</td>
<td></td>
<td>SUB MASC</td>
<td>Sub to MASC, Q=1, so data/2.</td>
</tr>
<tr>
<td>0798 D4 01DA</td>
<td></td>
<td>SUB CRLF</td>
<td>Output &quot;CR&quot; and &quot;LF&quot;</td>
</tr>
<tr>
<td>0799 B6 3E</td>
<td></td>
<td>LDN 3E</td>
<td>Load ASCII &quot;&gt;&quot; and output it</td>
</tr>
<tr>
<td>079D DA</td>
<td></td>
<td>SEP A</td>
<td>To show histogram change.</td>
</tr>
<tr>
<td>079E E5</td>
<td></td>
<td>RETURN</td>
<td>Return with Q still set for</td>
</tr>
<tr>
<td>079F 00</td>
<td></td>
<td>IDL</td>
<td>Auto histogram check.</td>
</tr>
</tbody>
</table>

The next set of routines computes the mark and space thresholds from the smoothed data. The main routine begins at 0600, but the subroutines are presented first.

Add. Code | Label | Mnemonic | Comments |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0740 F8 00</td>
<td>FMAX</td>
<td>LDI 00</td>
<td>Reset bin number</td>
</tr>
<tr>
<td>0742 AC</td>
<td></td>
<td>PLO C</td>
<td>And maximum value.</td>
</tr>
<tr>
<td>0743 BC</td>
<td></td>
<td>PHI C</td>
<td></td>
</tr>
<tr>
<td>0744 CD</td>
<td></td>
<td>LSQ</td>
<td>If right maximum set RF.0</td>
</tr>
<tr>
<td>0745 AF</td>
<td></td>
<td>PLO F</td>
<td>To 00, else leave it alone.</td>
</tr>
<tr>
<td>0746 C4</td>
<td></td>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>0747 E9</td>
<td>MAXM</td>
<td>SEX 9</td>
<td>Make R9 be RX for math use.</td>
</tr>
<tr>
<td>0748 9C</td>
<td></td>
<td>GHI C</td>
<td>Get current maximum and sub-</td>
</tr>
<tr>
<td>0749 F5</td>
<td></td>
<td>SD</td>
<td>Tract data pointed to by R9.</td>
</tr>
<tr>
<td>074A 3B 53</td>
<td></td>
<td>BL NCHG</td>
<td>If less, no change, so jump.</td>
</tr>
<tr>
<td>074C C5</td>
<td></td>
<td>LSNQ</td>
<td>Skip two if rightmost max.</td>
</tr>
<tr>
<td>074E E5</td>
<td>MAXM</td>
<td>SEX 9</td>
<td>That means change on 2&gt;. If leftmost max, change on 2 only</td>
</tr>
<tr>
<td>0750 BC</td>
<td></td>
<td>PHI C</td>
<td>Jump if leftmost and equal.</td>
</tr>
<tr>
<td>0751 8F</td>
<td></td>
<td>CLO F</td>
<td>If a change, get new maximum</td>
</tr>
<tr>
<td>0752 AC</td>
<td></td>
<td>PLO C</td>
<td>And put it in RC.1.</td>
</tr>
<tr>
<td>0753 F8</td>
<td></td>
<td>PLO C</td>
<td>Get new bin number and put</td>
</tr>
<tr>
<td>0754 00</td>
<td></td>
<td>IDL</td>
<td>It in RC.0.</td>
</tr>
</tbody>
</table>
MORSE PROGRAM

;This is a continuation of the "FMAX" subroutine.

<table>
<thead>
<tr>
<th>Add. Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0753 1F</td>
<td>NCHG</td>
<td>INC F</td>
<td>Increment bin number</td>
</tr>
<tr>
<td>0754 19</td>
<td>INC 9</td>
<td>; and memory pointer.</td>
<td></td>
</tr>
<tr>
<td>0755 89</td>
<td>GLO 9</td>
<td>; If R9=0600, mark scan done,</td>
<td></td>
</tr>
<tr>
<td>0756 32 5C</td>
<td>EZ MAXD</td>
<td>; Jump ahead.</td>
<td></td>
</tr>
<tr>
<td>0758 FF C0</td>
<td>SMI C0</td>
<td>; If R9=05BF, space scan done,</td>
<td></td>
</tr>
<tr>
<td>075A 3A 47</td>
<td>BNZ MAXM</td>
<td>; Else jump back for more.</td>
<td></td>
</tr>
<tr>
<td>075C E7</td>
<td>MAXD</td>
<td>SEX 7</td>
<td>; Restore R7 as RX</td>
</tr>
<tr>
<td>075D D5</td>
<td>RETURN</td>
<td>; and get out.</td>
<td></td>
</tr>
<tr>
<td>075E 00</td>
<td>IDL</td>
<td>; This next subroutine takes</td>
<td></td>
</tr>
<tr>
<td>075F 00</td>
<td>IDL</td>
<td>; The data passed from FMAX</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>; and sets it up for later use. Enter with Q=1 for marks.</td>
<td></td>
</tr>
<tr>
<td>0700 F8 00</td>
<td>ADD50</td>
<td>LDI 00</td>
<td>; Reset RF.0</td>
</tr>
<tr>
<td>0702 AF</td>
<td>PLO  F</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>0703 9C</td>
<td>GHI C</td>
<td>; Get maximum peak height.</td>
<td></td>
</tr>
<tr>
<td>0704 F6</td>
<td>SHR</td>
<td>; Divide by two</td>
<td></td>
</tr>
<tr>
<td>0705 BC</td>
<td>PHI C</td>
<td>; And put it back.</td>
<td></td>
</tr>
<tr>
<td>0706 F8 80</td>
<td>LDI 80</td>
<td>; Get space offset</td>
<td></td>
</tr>
<tr>
<td>0708 C5</td>
<td>LSNQ</td>
<td>; Skip 2 if spaces</td>
<td></td>
</tr>
<tr>
<td>0709 F8 C0</td>
<td>LDI C0</td>
<td>; Else get mark offset.</td>
<td></td>
</tr>
<tr>
<td>070B 57</td>
<td>STR  7</td>
<td>; Put offset in 07F0.</td>
<td></td>
</tr>
<tr>
<td>070C 8C</td>
<td>GLO C</td>
<td>; Get maximum bin number</td>
<td></td>
</tr>
<tr>
<td>070D F4</td>
<td>ADD</td>
<td>; Add offset for low address</td>
<td></td>
</tr>
<tr>
<td>070E A9</td>
<td>PLO  9</td>
<td>; And put it in R9.0.</td>
<td></td>
</tr>
<tr>
<td>070F D5</td>
<td>RETURN</td>
<td>;</td>
<td></td>
</tr>
</tbody>
</table>

; This subroutine finds the first occurrence of a bin |
; <= (maximum height)/2 while searching left to right. |
; Enter with start address of search in R9, (maximum)/2 |
; in RC, and 00 in RF.0. On exit RF.0 contains the number |
; of bins from the peak to <= (peak)/2. |

<table>
<thead>
<tr>
<th>Add. Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0710 00</td>
<td>IDL</td>
<td>; IDL's to 0713 inc.</td>
<td></td>
</tr>
<tr>
<td>0714 E9</td>
<td>LTOR</td>
<td>SEX 9</td>
<td>; Make R9 be RX</td>
</tr>
<tr>
<td>0715 9C</td>
<td>MOLR</td>
<td>GHI C</td>
<td>; Get 1/2 peak and</td>
</tr>
<tr>
<td>0716 F7</td>
<td>SM</td>
<td>; Subtract data.</td>
<td></td>
</tr>
<tr>
<td>0717 33 22</td>
<td>BCE LROK</td>
<td>; Out if bin found,</td>
<td></td>
</tr>
<tr>
<td>0719 19</td>
<td>INC 9</td>
<td>; Else increment pointer</td>
<td></td>
</tr>
<tr>
<td>071A 1F</td>
<td>INC F</td>
<td>; And bin offset.</td>
<td></td>
</tr>
<tr>
<td>071B 89</td>
<td>GLO 9</td>
<td>; If R9=0700</td>
<td></td>
</tr>
<tr>
<td>071C 32 22</td>
<td>BZ</td>
<td>; Get out, end of marks.</td>
<td></td>
</tr>
<tr>
<td>071E FF C0</td>
<td>SMI C0</td>
<td>; If R9=06C0, end of spaces,</td>
<td></td>
</tr>
<tr>
<td>0720 3A 15</td>
<td>BNZ MOLR</td>
<td>; Else jump back for more.</td>
<td></td>
</tr>
<tr>
<td>0722 E7</td>
<td>LROK</td>
<td>SEX 7</td>
<td>; Restore R7 as RX.</td>
</tr>
<tr>
<td>0723 D5</td>
<td>RETURN</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>0724 00</td>
<td>IDL</td>
<td>; IDL's to 0727 inc.</td>
<td></td>
</tr>
</tbody>
</table>
MORSE PROGRAM

;This subroutine is similar to the one above except that it searches from right to left. Q=0 for spaces and Q=1 for marks. RE.0 and RE.1 contain leftmost stop addresses.

<table>
<thead>
<tr>
<th>Add.</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0728</td>
<td>E9</td>
<td>RTOL</td>
<td>SEX 9</td>
<td>;Make R9 be RX</td>
</tr>
<tr>
<td>0729</td>
<td>9C</td>
<td>GHI</td>
<td>C</td>
<td>;Get 1/2 peak and subtract</td>
</tr>
<tr>
<td>072A</td>
<td>F7</td>
<td>SM</td>
<td></td>
<td>;data.</td>
</tr>
<tr>
<td>072B</td>
<td>33</td>
<td>BGE</td>
<td>RLOK</td>
<td>;Out if bin found,</td>
</tr>
<tr>
<td>072D</td>
<td>E7</td>
<td>SEX</td>
<td>7</td>
<td>;else restore R7 as RX,</td>
</tr>
<tr>
<td>072E</td>
<td>89</td>
<td>GLO</td>
<td>9</td>
<td>;Get memory pointer and</td>
</tr>
<tr>
<td>072F</td>
<td>FF</td>
<td>SMI</td>
<td>80</td>
<td>;subtract spaces offset,</td>
</tr>
<tr>
<td>0731</td>
<td>C5</td>
<td>LSNQ</td>
<td></td>
<td>;Skip 2 if spaces, else</td>
</tr>
<tr>
<td>0732</td>
<td>FF</td>
<td>SMI</td>
<td>40</td>
<td>;subtract extra mark offset,</td>
</tr>
<tr>
<td>0734</td>
<td>57</td>
<td>STR</td>
<td>7</td>
<td>;Stuff it in 07F0</td>
</tr>
<tr>
<td>0735</td>
<td>8E</td>
<td>GLO</td>
<td>E</td>
<td>;Get overrange for spaces,</td>
</tr>
<tr>
<td>0736</td>
<td>C5</td>
<td>LSNQ</td>
<td></td>
<td>;skip 2 if spaces,</td>
</tr>
<tr>
<td>0737</td>
<td>9E</td>
<td>GHI</td>
<td>E</td>
<td>;else get it for marks,</td>
</tr>
<tr>
<td>0738</td>
<td>C4</td>
<td>NOP</td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>0739</td>
<td>F7</td>
<td>SM</td>
<td></td>
<td>;Compare pointer and overrange</td>
</tr>
<tr>
<td>073A</td>
<td>29</td>
<td>DEC</td>
<td>9</td>
<td>;Decrement memory pointer,</td>
</tr>
<tr>
<td>073B</td>
<td>1F</td>
<td>INC</td>
<td>F</td>
<td>;Increment bin counter,</td>
</tr>
<tr>
<td>073C</td>
<td>3B</td>
<td>BM</td>
<td>RTOL</td>
<td>;If minus, back for more,</td>
</tr>
<tr>
<td>073E</td>
<td>E7</td>
<td>RLOK</td>
<td>SEX 7</td>
<td>;else restore RX</td>
</tr>
<tr>
<td>073F</td>
<td>D5</td>
<td>RETURN</td>
<td></td>
<td>;and get out.</td>
</tr>
</tbody>
</table>

;This is the routine to calculate the mark threshold. It calls the above three routines, FMAX, LTOR, & RTOL. In the first part "peak" or "maximum" refers to the histogram peak generated by dots.

<table>
<thead>
<tr>
<th>Add.</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0600</td>
<td>F8</td>
<td>06</td>
<td>THRM</td>
<td>LDI 06</td>
</tr>
<tr>
<td>0602</td>
<td>B9</td>
<td></td>
<td>PHI 9</td>
<td></td>
</tr>
<tr>
<td>0603</td>
<td>F8</td>
<td>C0</td>
<td>LDI C0</td>
<td></td>
</tr>
<tr>
<td>0605</td>
<td>A9</td>
<td></td>
<td>PLO 9</td>
<td></td>
</tr>
<tr>
<td>0606</td>
<td>7A</td>
<td>REQ</td>
<td></td>
<td>;Reset Q, shows right maximum</td>
</tr>
<tr>
<td>0607</td>
<td>D4</td>
<td>0740</td>
<td>SUB</td>
<td>FMAX</td>
</tr>
<tr>
<td>060A</td>
<td>7B</td>
<td>SEQ</td>
<td></td>
<td>;Set Q, shows marks.</td>
</tr>
<tr>
<td>060B</td>
<td>F8</td>
<td>06</td>
<td>LDI 06</td>
<td></td>
</tr>
<tr>
<td>060D</td>
<td>B9</td>
<td>PHI</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>060E</td>
<td>D4</td>
<td>0700</td>
<td>SUB</td>
<td>ADD50</td>
</tr>
<tr>
<td>0611</td>
<td>D4</td>
<td>0714</td>
<td>SUB</td>
<td>LTOR</td>
</tr>
</tbody>
</table>
MORSE PROGRAM

;This is the continuation of the mark threshold routine.
<table>
<thead>
<tr>
<th>Add. Code</th>
<th>Label Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0614 5F</td>
<td>GLO F</td>
<td>Get distance from peak</td>
</tr>
<tr>
<td>0615 57</td>
<td>STR 7</td>
<td>and stuff into 07F0.</td>
</tr>
<tr>
<td>0616 5C</td>
<td>GLO C</td>
<td>Get peak bin # and</td>
</tr>
<tr>
<td>0617 F4</td>
<td>ADD</td>
<td>find bin twice as far from</td>
</tr>
<tr>
<td>0618 F4</td>
<td>ADD</td>
<td>peak as 50% bin.</td>
</tr>
<tr>
<td>0619 BE</td>
<td>PHI E</td>
<td>Put it in RE.1 as left stop</td>
</tr>
<tr>
<td>061A AF</td>
<td>PLO F</td>
<td>and also into RF.0.</td>
</tr>
<tr>
<td>061B FC</td>
<td>ADI C0</td>
<td>Add offsets to make this</td>
</tr>
<tr>
<td>061C A9</td>
<td>PLO 9</td>
<td>a memory address in R9 in</td>
</tr>
<tr>
<td>061D F8 06</td>
<td>LDI 06</td>
<td>the smoothed mark range.</td>
</tr>
<tr>
<td>0620 B9</td>
<td>PHI 9</td>
<td></td>
</tr>
</tbody>
</table>

;From this point "peak" refers to the dash cluster.

<table>
<thead>
<tr>
<th>Add. Code</th>
<th>Label Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0621 7B</td>
<td>SEQ</td>
<td>Set Q to find left maximum.</td>
</tr>
<tr>
<td>0622 D4 0740</td>
<td>SUB FMAX</td>
<td>Sub to find left maximum.</td>
</tr>
<tr>
<td>0625 F8 06</td>
<td>LDI 06</td>
<td>Put partial address in R9.1.</td>
</tr>
<tr>
<td>0627 B9</td>
<td>PHI 9</td>
<td></td>
</tr>
<tr>
<td>0628 7B</td>
<td>SEQ</td>
<td>Set Q for marks.</td>
</tr>
<tr>
<td>0629 D4 0700</td>
<td>SUB ADD50</td>
<td>Sub to fix data.</td>
</tr>
<tr>
<td>062C D4 0728</td>
<td>SUB RTOL</td>
<td>Sub to find 50% peak bin.</td>
</tr>
<tr>
<td>062F 5F</td>
<td>GLO F</td>
<td>Get distance from peak,</td>
</tr>
<tr>
<td>0630 FE</td>
<td>SHL</td>
<td>double it</td>
</tr>
<tr>
<td>0631 57</td>
<td>STR 7</td>
<td>and stuff it in 07F0.</td>
</tr>
<tr>
<td>0632 5C</td>
<td>GLO C</td>
<td>Get peak bin # and find</td>
</tr>
<tr>
<td>0633 F7</td>
<td>SM</td>
<td>bin twice as far from peak</td>
</tr>
<tr>
<td>0634 57</td>
<td>STR 7</td>
<td>as 50% and stuff it in 07F0.</td>
</tr>
<tr>
<td>0635 9E</td>
<td>CHI E</td>
<td>Add same result from dot</td>
</tr>
<tr>
<td>0636 F4</td>
<td>ADD</td>
<td>peak and divide by two to</td>
</tr>
<tr>
<td>0637 F6</td>
<td>SHR</td>
<td>find estimate for mark</td>
</tr>
<tr>
<td>0638 BE</td>
<td>PHI E</td>
<td>threshold, put it in RE.1.</td>
</tr>
<tr>
<td>0639 C0 0640</td>
<td>LBR THRS</td>
<td>Jump for space threshold.</td>
</tr>
<tr>
<td>063C 00</td>
<td>IDL</td>
<td>IDL's to 063F inc.</td>
</tr>
</tbody>
</table>

;This is the routine to calculate the space threshold.
;In the first part "peak" or "maximum", refers to the 
;histogram peak generated by the intra-character spaces.

<table>
<thead>
<tr>
<th>Add. Code</th>
<th>Label Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0640 F8 06</td>
<td>THRS LDI 06</td>
<td>0680 into R9, start address</td>
</tr>
<tr>
<td>0642 B9</td>
<td>PHI 9</td>
<td>of smoothed space data.</td>
</tr>
<tr>
<td>0643 F8 80</td>
<td>LDI 80</td>
<td></td>
</tr>
<tr>
<td>0645 A9</td>
<td>PLO 9</td>
<td></td>
</tr>
<tr>
<td>0646 7A</td>
<td>REQ</td>
<td>Reset Q, shows right maximum</td>
</tr>
<tr>
<td>0647 D4 0740</td>
<td>SUB FMAX</td>
<td>Sub to find right maximum.</td>
</tr>
<tr>
<td>064A 7A</td>
<td>REQ</td>
<td>Reset Q, shows spaces.</td>
</tr>
<tr>
<td>064B F8 06</td>
<td>LDI 06</td>
<td>Put partial address in R9.1.</td>
</tr>
<tr>
<td>064D B9</td>
<td>PHI 9</td>
<td></td>
</tr>
</tbody>
</table>
MORSE PROGRAM

;This is the continuation of the space threshold routine.

<table>
<thead>
<tr>
<th>Add.</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>064E</td>
<td>D4</td>
<td>0700</td>
<td>SUB</td>
<td>Add to fix data.</td>
</tr>
<tr>
<td>0651</td>
<td>D4</td>
<td>0714</td>
<td>SUB</td>
<td>Sub to find 50% peak bin.</td>
</tr>
<tr>
<td>0654</td>
<td>8F</td>
<td></td>
<td>GLO</td>
<td>Get distance from peak</td>
</tr>
<tr>
<td>0655</td>
<td>57</td>
<td></td>
<td>STR</td>
<td>and stuff into 07FO.</td>
</tr>
<tr>
<td>0656</td>
<td>8C</td>
<td></td>
<td>GLO</td>
<td>Get peak bin # and</td>
</tr>
<tr>
<td>0657</td>
<td>F4</td>
<td></td>
<td>ADD</td>
<td>find bin twice as far from</td>
</tr>
<tr>
<td>0658</td>
<td>F4</td>
<td></td>
<td>ADD</td>
<td>peak as 50% bin.</td>
</tr>
<tr>
<td>0659</td>
<td>AE</td>
<td></td>
<td>PLO</td>
<td>Put it in RE.0 as left stop</td>
</tr>
<tr>
<td>065A</td>
<td>AF</td>
<td></td>
<td>PLO</td>
<td>and also into RF.0.</td>
</tr>
<tr>
<td>065B</td>
<td>FC</td>
<td>80</td>
<td>ADI</td>
<td>Add offsets to make this</td>
</tr>
<tr>
<td>065D</td>
<td>A9</td>
<td></td>
<td>LDI</td>
<td>a memory address in R9 in</td>
</tr>
<tr>
<td>065E</td>
<td>F8</td>
<td>06</td>
<td>LDI</td>
<td>the smoothed space range.</td>
</tr>
<tr>
<td>0660</td>
<td>B9</td>
<td></td>
<td>PHI</td>
<td>;</td>
</tr>
</tbody>
</table>

;Here "peak" refers to the inter-character space cluster.

<table>
<thead>
<tr>
<th>Add.</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0661</td>
<td>7B</td>
<td></td>
<td>SEQ</td>
<td>Set Q to find left maximum.</td>
</tr>
<tr>
<td>0662</td>
<td>D4</td>
<td>0740</td>
<td>SUB</td>
<td>Sub to find left maximum.</td>
</tr>
<tr>
<td>0663</td>
<td>F8</td>
<td>06</td>
<td>LDI</td>
<td>Put partial address in R9.1.</td>
</tr>
<tr>
<td>0667</td>
<td>B9</td>
<td></td>
<td>PHI</td>
<td>;</td>
</tr>
<tr>
<td>0668</td>
<td>7A</td>
<td></td>
<td>REQ</td>
<td>Reset Q for spaces.</td>
</tr>
<tr>
<td>0669</td>
<td>D4</td>
<td>0700</td>
<td>SUB</td>
<td>Sub to fix data.</td>
</tr>
<tr>
<td>066C</td>
<td>D4</td>
<td>0728</td>
<td>SUB</td>
<td>Sub to find 50% peak bin.</td>
</tr>
<tr>
<td>066F</td>
<td>8F</td>
<td></td>
<td>GLO</td>
<td>Get distance from peak,</td>
</tr>
<tr>
<td>0670</td>
<td>FE</td>
<td></td>
<td>SHL</td>
<td>double it</td>
</tr>
<tr>
<td>0671</td>
<td>57</td>
<td></td>
<td>STR</td>
<td>and stuff it in 07FO.</td>
</tr>
<tr>
<td>0672</td>
<td>8C</td>
<td></td>
<td>GLO</td>
<td>Get peak bin # and find</td>
</tr>
<tr>
<td>0673</td>
<td>F7</td>
<td></td>
<td>SM</td>
<td>bin twice as far from peak</td>
</tr>
<tr>
<td>0674</td>
<td>57</td>
<td></td>
<td>STR</td>
<td>as 50% and stuff it in 07FO.</td>
</tr>
<tr>
<td>0675</td>
<td>8E</td>
<td></td>
<td>GLO</td>
<td>Add result from intra-space</td>
</tr>
<tr>
<td>0676</td>
<td>F4</td>
<td></td>
<td>ADD</td>
<td>peak and divide by two for</td>
</tr>
<tr>
<td>0677</td>
<td>F6</td>
<td></td>
<td>SHR</td>
<td>first estimate of space</td>
</tr>
<tr>
<td>0678</td>
<td>57</td>
<td></td>
<td>STR</td>
<td>threshold, put it in 07FO.</td>
</tr>
</tbody>
</table>

;Due the poor "quality" of the inter-character space peak, |
it is averaged with the mark threshold which would be the |
;same for perfect code. |

<table>
<thead>
<tr>
<th>Add.</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0679</td>
<td>9E</td>
<td></td>
<td>CHI</td>
<td>Get mark threshold and</td>
</tr>
<tr>
<td>067A</td>
<td>F4</td>
<td></td>
<td>ADD</td>
<td>simply average with estimate</td>
</tr>
<tr>
<td>067B</td>
<td>F6</td>
<td></td>
<td>SHR</td>
<td>;just calculated. Store this</td>
</tr>
<tr>
<td>067C</td>
<td>AE</td>
<td></td>
<td>PLO</td>
<td>final result in RE.0</td>
</tr>
<tr>
<td>067D</td>
<td>C0</td>
<td>07B0</td>
<td>LBR CLUP</td>
<td>Jump to cleanup loose ends.</td>
</tr>
</tbody>
</table>
MORSE PROGRAM

;This short routine follows the space threshold calculation
to clear up various odds and ends.

Add.  Code  Label  Mnemonic  Comments
07A0  00  IDL  IDL's to 07AF inc.
07B0  D4  01AB CLUP  SUB DISPY  ;Display new thresholds.
07B3  F8  00  LDI  00  ;
07B5  A9  PLO  9  ;Reset key down timer and
counter for key downs in
07B6  BC  PHI  C  ;character, Set Morse shift
07B7  F8  01  LDI  01  ;register to 01.
07B9  B9  PHI  9  ;Point to stored RC.0,
07BA  27  DEC  7  ;get it, point to 07F0,
07BB  72  LDAXA  ;restore RC.0.
07BC  AC  PLO  C
07BD  C0  0400  LBR TIME  ;Jump back to timer.

;These are the commands which the program recognizes from
;an ASCII keyboard. They are immediately recognized, so
:no carriage return is required.

/  Restart at 03DD, sub to monitor to enter new speed,
pres "R" to return.
H  Output histogram immediately, wait for more commands.
K  Send hardcopy command to graphics terminal, wait for
more commands.
M  Restart at 03F6, same histogram, speed and thresholds.
N  Cancel autohistograms, continue at 0400.
X  Zero raw histogram data, restart at 03F6.
Y  Divide raw histogram data by two, restart at 03F6.
Z  Set automatic mode for histograms. If any raw bin is
full, the program stops and outputs a histogram. The
raw data is divided by two, but until the program is
restarted the smoothed data is untouched, so other
copies can be made. This command continues at 0400.

"N" or "Z" can be pressed at any time during normal
program execution. The other commands all disrupt
decoding. Any command not in the above table will
cause a restart of the entire system to the monitor.
MORSE PROGRAM

;The above routines were all essential to decoding Morse code. The remaining sections are only needed to output histograms to the Tektronix graphics terminal. Register A points to the routine for serial port 2, not 1 as usual. A lot of manipulation is necessary because the plotter works on a 1024X by 779Y absolute matrix, with each letter 14X by 22Y, and the histogram requires 64X by 31Y numbered positions, each divisible by four!!! Calculations can only be done in integer math, so there are problems. This subroutine numbers an axis 0 to F repeatedly.

<table>
<thead>
<tr>
<th>Add.</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0220</td>
<td>F8 00</td>
<td>LTTR</td>
<td>LDI 00</td>
<td>;00 into RF.0 if not Q,</td>
</tr>
<tr>
<td>0222</td>
<td>C5</td>
<td></td>
<td>LSNQ</td>
<td>;0F if Q, start number of</td>
</tr>
<tr>
<td>0223</td>
<td>F8 0F</td>
<td></td>
<td>LDI 0F</td>
<td>;hex-numbered axis,</td>
</tr>
<tr>
<td>0225</td>
<td>AF</td>
<td></td>
<td>PLO F</td>
<td>;</td>
</tr>
<tr>
<td>0226</td>
<td>F8 1D</td>
<td>MLTR?</td>
<td>LDI 1D</td>
<td>;Output &quot;GS&quot; to set</td>
</tr>
<tr>
<td>0228</td>
<td>DA</td>
<td></td>
<td>SEP A</td>
<td>;graphics mode,</td>
</tr>
<tr>
<td>0229</td>
<td>D4 02E0</td>
<td></td>
<td>SUB XYOUT1</td>
<td>;Output XY coordinates,</td>
</tr>
<tr>
<td>022C</td>
<td>D4 02F8</td>
<td></td>
<td>SUB XYOUT2</td>
<td></td>
</tr>
<tr>
<td>022F</td>
<td>F8 1F</td>
<td></td>
<td>LDI 1F</td>
<td>;Output &quot;US&quot; to set</td>
</tr>
<tr>
<td>0231</td>
<td>DA</td>
<td></td>
<td>SEP A</td>
<td>;alphaneumeric mode,</td>
</tr>
<tr>
<td>0232</td>
<td>8F</td>
<td></td>
<td>CLO F</td>
<td>;Get axis counter, mask in</td>
</tr>
<tr>
<td>0233</td>
<td>FA 0F</td>
<td></td>
<td>ANI 0F</td>
<td>;lower 4 bits, put them</td>
</tr>
<tr>
<td>0235</td>
<td>BF</td>
<td></td>
<td>PHI F</td>
<td>;in RF.1 and output them as</td>
</tr>
<tr>
<td>0239</td>
<td>73</td>
<td></td>
<td>STXD</td>
<td>;a single hex digit.</td>
</tr>
<tr>
<td>0240</td>
<td>F8 F0</td>
<td></td>
<td>LDI F0</td>
<td>;Point RX back to 07F0,</td>
</tr>
<tr>
<td>0242</td>
<td>A7</td>
<td></td>
<td>PLO 7</td>
<td>;RX points to coordinate,</td>
</tr>
<tr>
<td>023B</td>
<td>9C</td>
<td></td>
<td>CLO C</td>
<td>;Get coordinate increment,</td>
</tr>
<tr>
<td>023C</td>
<td>F4</td>
<td></td>
<td>ADD</td>
<td>;add it on,</td>
</tr>
<tr>
<td>023D</td>
<td>73</td>
<td></td>
<td>STXD</td>
<td>;restore coordinate,</td>
</tr>
<tr>
<td>023E</td>
<td>F8 F0</td>
<td></td>
<td>LDI F0</td>
<td>;Point RX back to 07F0,</td>
</tr>
<tr>
<td>0240</td>
<td>A7</td>
<td></td>
<td>PLO 7</td>
<td>;</td>
</tr>
<tr>
<td>0241</td>
<td>:F</td>
<td></td>
<td>INC F</td>
<td>;Increment axis counter, but</td>
</tr>
<tr>
<td>0242</td>
<td>C5</td>
<td></td>
<td>LSNQ</td>
<td>;if Q was set, decrement</td>
</tr>
<tr>
<td>0243</td>
<td>2F</td>
<td></td>
<td>DEC F</td>
<td>;counter by 2, effectively</td>
</tr>
<tr>
<td>0244</td>
<td>2F</td>
<td></td>
<td>DEC F</td>
<td>;by 1,</td>
</tr>
<tr>
<td>0245</td>
<td>8F</td>
<td></td>
<td>CLO F</td>
<td>;Get axis counter and</td>
</tr>
<tr>
<td>0246</td>
<td>57</td>
<td></td>
<td>STR 7</td>
<td>;stuff it in 07F0,</td>
</tr>
<tr>
<td>0247</td>
<td>89</td>
<td></td>
<td>GLO 9</td>
<td>;Get stop count and</td>
</tr>
<tr>
<td>0248</td>
<td>F7</td>
<td></td>
<td>SM</td>
<td>;subtract,</td>
</tr>
<tr>
<td>0249</td>
<td>3A 25</td>
<td></td>
<td>BNZ MLTR?</td>
<td>;jump back for more if not</td>
</tr>
<tr>
<td>024B</td>
<td>D5</td>
<td></td>
<td>RETURN</td>
<td>;equal, else return.</td>
</tr>
<tr>
<td>024C</td>
<td>00</td>
<td></td>
<td>IDL</td>
<td>;IDL's to 024F inc.</td>
</tr>
</tbody>
</table>
MORSE PROGRAM

;These two subroutines output an XY coordinate to the
;graphics terminal which requires 10 bit X and Y values,
broken into four 5 bit ASCII characters. Enter with an
;8 bit X value in 07F1 and Y in 07F2. Normally call XYOU2
;immediately after XYOU1, but to obtain the full 10 bit
;precision available, the two lsb's in the Y value can be
;added at that point.

Add. Code Label Mnemonic Comments
02E0 60 XYOU1 IRX ;Point to X value,
02E1 72 LDXA ;get it and point to Y.
02E2 A0 PLO 0 ;X into R0.0,
02E3 F0 LDX ;get Y,
02E4 B0 PHI 0 ;Y into R0.1,
02E5 27 DEC 7 ;Point to 07F0.
02E6 27 DEC 7 ;
02E7 F6 SHR ;Get high 5 Y bits in lower
02E8 F6 SHR 3 positions.
02E9 F6 SHR ;
02EA F9 20 ORI 20 ;OR in identifier and
02EC DA SEP A ;output high Y byte.
02ED 90 CHI 0 ;Get Y again,
02EE FE SHL ;put 0's in two lsb's
02EF FE SHL ;and stuff it in 07F0.
02F0 57 STR 7 ;
02F1 D5 RETURN ;Return for possible mods to
;two lsb's to get 10 bit Y coordinate if required.
02F2 00 IDL ;IDL's to 02F7 inc.
02F8 F0 XYOU2 LDX ;Get Y value back,
02F9 FA 1F ANI 1F ;mask in lower 5 bits only
02FB F9 60 ORI 60 ;OR in identifier and
02FD DA SEP A ;output low Y byte.
02FE 80 CLO 0 ;Get X value and
02FF F6 SHR ;put high 5 bits in lower
0300 F6 SHR 5 positions.
0301 F6 SHR ;
0302 F9 20 ORI 20 ;OR in identifier and
0304 DA SEP A ;output high X byte.
0305 80 CLO 0 ;Get X again and
0306 FE SHL ;put 0's in two lsb's.
0307 FE SHL ;
0308 FA 1C ANI 1C ;Mask in required bits,
030A F9 40 ORI 40 ;OR in identifier and
030C DA SEP A ;output low X byte.
030D D5 RETURN ;All done, get out.
030E 00 IDL ;
030F 00 IDL ;
The above three subroutines are called by this section which handles the overhead for lettering the axes. This is the actual start of the histogram routine.

<table>
<thead>
<tr>
<th>Add.</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0250</td>
<td>D4</td>
<td>01DA</td>
<td>HIST</td>
<td>SUB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CRLF</td>
</tr>
<tr>
<td>0253</td>
<td>F8</td>
<td>10</td>
<td>LDI</td>
<td>Point RA to output routine</td>
</tr>
<tr>
<td></td>
<td>AA</td>
<td></td>
<td>PLO</td>
<td>for serial port 2.</td>
</tr>
<tr>
<td>0256</td>
<td>F8</td>
<td>07</td>
<td>LDI</td>
<td>Output &quot;BEL&quot; to inform</td>
</tr>
<tr>
<td></td>
<td>DA</td>
<td></td>
<td>SEP</td>
<td>operator of histogram.</td>
</tr>
<tr>
<td>0259</td>
<td>F8</td>
<td>1B</td>
<td>LDI</td>
<td>Output &quot;ESC&quot; &amp; &quot;FF&quot; to</td>
</tr>
<tr>
<td></td>
<td>BA</td>
<td></td>
<td>SEP</td>
<td>erase screen.</td>
</tr>
<tr>
<td>025C</td>
<td>F8</td>
<td>0C</td>
<td>LDI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FA</td>
<td></td>
<td>SEP</td>
<td></td>
</tr>
<tr>
<td>025F</td>
<td>FF</td>
<td></td>
<td>LDI</td>
<td>This is a delay while</td>
</tr>
<tr>
<td></td>
<td>FF</td>
<td></td>
<td>PHI</td>
<td>the screen clears.</td>
</tr>
<tr>
<td>0261</td>
<td>AF</td>
<td></td>
<td>PLO</td>
<td>FFFFF into RF</td>
</tr>
<tr>
<td>0262</td>
<td>2F</td>
<td></td>
<td>SCRP</td>
<td>DEC F</td>
</tr>
<tr>
<td>0264</td>
<td>9F</td>
<td></td>
<td>GHI</td>
<td>Get RF.1, jump back if not</td>
</tr>
<tr>
<td>0265</td>
<td>3A</td>
<td>63</td>
<td>BNZ SCRP</td>
<td>zero, else delay over.</td>
</tr>
<tr>
<td>0267</td>
<td>F8</td>
<td>F1</td>
<td>LDI</td>
<td>Set up to label horizontal</td>
</tr>
<tr>
<td></td>
<td>AC</td>
<td></td>
<td>PLO</td>
<td>axis (see LTTR for expl.)</td>
</tr>
<tr>
<td>026A</td>
<td>F8</td>
<td>04</td>
<td>LDI</td>
<td>F1 in RC.0 = X axis</td>
</tr>
<tr>
<td></td>
<td>BC</td>
<td></td>
<td>PHI</td>
<td>04 in RC.1 = increment</td>
</tr>
<tr>
<td>026D</td>
<td>57</td>
<td></td>
<td>IRX</td>
<td>04 in 07F1 = X offset</td>
</tr>
<tr>
<td>026E</td>
<td>60</td>
<td></td>
<td>STR</td>
<td>57 in 07F2 = Y offset</td>
</tr>
<tr>
<td>026F</td>
<td>60</td>
<td></td>
<td>IRX</td>
<td>3F in R9.0 = last count+1</td>
</tr>
<tr>
<td>0270</td>
<td>F8</td>
<td>61</td>
<td>LDI</td>
<td>0 in Q = pos. incr.</td>
</tr>
<tr>
<td></td>
<td>73</td>
<td></td>
<td>STXD</td>
<td></td>
</tr>
<tr>
<td>0272</td>
<td>27</td>
<td></td>
<td>DEC</td>
<td>Point to 07F0 again.</td>
</tr>
<tr>
<td>0274</td>
<td>F8</td>
<td>3F</td>
<td>LDI</td>
<td></td>
</tr>
<tr>
<td>0276</td>
<td>A9</td>
<td></td>
<td>PLO</td>
<td></td>
</tr>
<tr>
<td>0277</td>
<td>7A</td>
<td></td>
<td>SEQ</td>
<td></td>
</tr>
<tr>
<td>027B</td>
<td>D4</td>
<td>0220</td>
<td>SUB</td>
<td>Go and letter X axis.</td>
</tr>
<tr>
<td></td>
<td>F8</td>
<td>FF</td>
<td>LDI</td>
<td>F2 in RC.0 = Y axis</td>
</tr>
<tr>
<td>027D</td>
<td>A9</td>
<td></td>
<td>PLO</td>
<td>05 in RC.1 = increment</td>
</tr>
<tr>
<td>027E</td>
<td>F8</td>
<td>F2</td>
<td>LDI</td>
<td>00 in 07F1 = X offset</td>
</tr>
<tr>
<td></td>
<td>AC</td>
<td></td>
<td>PLO</td>
<td>11 in 07F2 = Y offset</td>
</tr>
<tr>
<td>0281</td>
<td>F8</td>
<td>05</td>
<td>LDI</td>
<td>FF in R9.0 = last count-1</td>
</tr>
<tr>
<td></td>
<td>BC</td>
<td></td>
<td>PHI</td>
<td>1 in Q = neg. incr.</td>
</tr>
<tr>
<td>0284</td>
<td>F8</td>
<td>11</td>
<td>LDI</td>
<td></td>
</tr>
<tr>
<td>0286</td>
<td>60</td>
<td></td>
<td>IRX</td>
<td></td>
</tr>
<tr>
<td>0287</td>
<td>60</td>
<td></td>
<td>IRX</td>
<td></td>
</tr>
<tr>
<td>0288</td>
<td>73</td>
<td></td>
<td>STXD</td>
<td></td>
</tr>
<tr>
<td>0289</td>
<td>F8</td>
<td>00</td>
<td>LDI</td>
<td></td>
</tr>
<tr>
<td>028B</td>
<td>73</td>
<td></td>
<td>STXD</td>
<td>Point to 07F0 again.</td>
</tr>
<tr>
<td>028C</td>
<td>7B</td>
<td></td>
<td>SEQ</td>
<td></td>
</tr>
<tr>
<td>028D</td>
<td>D4</td>
<td>0220</td>
<td>SUB</td>
<td>Letter Y axis from F to 0.</td>
</tr>
</tbody>
</table>
Add.  Code  Label  Mnemonic  Comments
0290  F8  10  LDI  10  ;Continue lettering axes.
0292  A9  PLO  9  ;F2 in RC.0 = Y axis
0293  60  IRX  ;05 in RC.1 = increment
0294  60  IRX  ;00 in 07F1 = X offset
0295  F8  66  LDI  66  ;66 in 07F2 = Y offset
0297  73  STXD  ;10 in R9.0 = last count+1
0298  F8  00  LDI  00  ; 0 in Q = pos. incr.
0299  73  STXD  ;Point to 07F0 again.
029B  C4  NOP  ;
029C  7A  REQ  ;
029D  D4  0220  SUB  LTTR  ;Letter Y axis from 0 to F.
02A0  C0  0310  LBR  TEKH  ;Continue with histogram.
02A3  00  IDL  ;IDL's to 02AF inc.
;This section outputs the data to the histogram.
;The smoothed data is used, but this is easily changed.
0310  F8  06  TEKH  LDI  06  ;Enter with Q=0, spaces.
0312  BC  PHI  C  ;0680 in RC = spaces, if Q=0
0313  60  IRX  ;06C0 in RC = marks, if Q=1
0314  F8  05  LDI  05  ;05 in 07F1 = X offset
0316  57  STR  7  ;60 in 07F2 = Y offset, Q=0
0317  60  IRX  ;66 in 07F2 = Y offset, Q=1
0318  F8  80  LDI  80  ;
031A  C5  LSNQ  ;
031D  AC  PLO  C  ;
031E  F8  60  TEK1  LDI  60  ;
0320  C5  LSNQ  ;
0321  F8  66  LDI  66  ;
0323  73  STXD  ;
0324  27  DEC  7  ;Point to 07F0 again.
0325  C4  NOP  ;
0326  F8  1D  LDI  1D  ;Output "CS", make this dark
0328  DA  SEP  A  ;vector to new position.
0329  D4  02E0  SUB  XYOU1  ;Output coordinates.
032C  D4  02F8  SUB  XYOU2  ;
032F  60  IRX  ;
0330  60  IRX  ;Point R7 to Y coordinate.
0331  C4  NOP  ;
0332  0C  LDN  C  ;Get smoothed data pointed
0333  F6  SHR  ;to by RC,
0334  F6  SHR  ;divide by 4,
0335  EC  SEX  C  ;
0336  F4  ADD  ;add that to original,
0337  E7  SEX  7  ;in effect multiply by 1.25.
MORSE PROGRAM

; In order to fit histogram onto screen with even spacing, 
; the very last (3Fth) bin is not output. This is not a 
; problem as there should be no useful information there.

Add. Code Label Mnemonic Comments

0338 CD LSQ ; Skip 2 if Q=1, marks.  
0339 F5 SD ; Spaces, vector down, so  
033A C4 NOP ; subtract data from offset.  
033B C5 LSNQ ; Skip 2 if Q=0, spaces.  
033C F4 ADD ; Marks, vector up, so add  
033D C4 NOP ; data to offset.  
033E 73 STXD ; Store Y coordinate in 07F2,  
033F 27 DEC 7 ; point back to 07F0.  
0340 C4 NOP ;  
0341 D4 02E0 SUB XYOU1 ; Sub to output high Y byte.  
0344 0C LDN C ; Get data again,  
0345 FA 03 ANI 03 ; mask in lower 3 bits only.  
0347 CD LSQ ; Skip 2 if Q = 1 = marks,  
0348 F5 SD ; else spaces, subtract.  
0349 C4 NOP ;  
034A C5 LSNQ ; Skip 2 if Q = 0 = spaces,  
034B F4 ADD ; else marks, add. Store  
034C C4 NOP ; this result back in 07F2,  
034D 57 STR 7 ; This adds the two lsb's to  
034E D4 02F8 SUB XYOU2 ; Y for 10 bits, output it.  
0351 1C INC C ; Increment RC, point to next  
0352 60 IRX ; data, increment RX,  
0353 F0 LDX ; get X coordinate.  
0354 FC 04 ADI 04 ; Add 04, move along axis,  
0355 57 STR 7 ; restore it.  
0356 60 IRX ; Point RX to Y coordinate.  
0358 8C GLO C ; Get low data pointer,  
0359 FF BF SMI BF ; subtract BF, neg if still  
035B CF LSDF ; spaces, skip 2 if positive.  
035C 30 1E BR TEK1 ; Neg, jump back, more spaces  
035E 7B SEQ ; Set Q for marks,  
035F 27 DEC 7 ; point to 07F0 again.  
0360 27 DEC 7 ; If above subtraction was 0,  
0361 32 10 EZ TEKH ; back to START mark output.  
0362 60 IRX ; Else point to 07F2 = Y.  
0363 60 IRX ;  
0364 FF 40 SMI 40 ; Subtract 40 from previous  
0365 3A 1E BNZ TEK1 ; result, if nonzero, back  
0366 7A REQ ; for more spaces, else reset  
0367 27 DEC 7 ; Q, point back to 07F0,  
036B 27 DEC 7 ;  
036C C0 0370 LBR OUPAR ; and jump ahead to output  
036D 00 IDL ; parameters.
### MORSE PROGRAM

This section outputs the speed and thresholds to the graphics terminal at the top of the histogram.

<table>
<thead>
<tr>
<th>Add.</th>
<th>Code</th>
<th>Label</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0370</td>
<td>60</td>
<td>OUPAR</td>
<td>IRX</td>
<td>;00 in 07F1 = X offset</td>
</tr>
<tr>
<td>0371</td>
<td>60</td>
<td></td>
<td>IRX</td>
<td>;BA in 07F2 = Y offset</td>
</tr>
<tr>
<td>0372</td>
<td>F8</td>
<td>BA</td>
<td>LDI BA</td>
<td></td>
</tr>
<tr>
<td>0374</td>
<td>73</td>
<td></td>
<td>STXD</td>
<td></td>
</tr>
<tr>
<td>0375</td>
<td>F8</td>
<td>00</td>
<td>LDI 00</td>
<td></td>
</tr>
<tr>
<td>0377</td>
<td>73</td>
<td></td>
<td>STXD</td>
<td>;Point back to 07F0.</td>
</tr>
<tr>
<td>0378</td>
<td>C4</td>
<td></td>
<td>NOP</td>
<td>;NOP's to 037B inc.</td>
</tr>
<tr>
<td>037C</td>
<td>FB</td>
<td>1D</td>
<td>LDI 1D</td>
<td>;Output &quot;GS&quot; for dark vector</td>
</tr>
<tr>
<td>037E</td>
<td>DA</td>
<td></td>
<td>SEP A</td>
<td></td>
</tr>
<tr>
<td>037F</td>
<td>D4</td>
<td>02E0</td>
<td>SUB XYOU1</td>
<td>;Output coordinates for</td>
</tr>
<tr>
<td>0382</td>
<td>D4</td>
<td>02F8</td>
<td>SUB XYOU2</td>
<td>;first lettering position.</td>
</tr>
<tr>
<td>0385</td>
<td>F8</td>
<td>1F</td>
<td>LDI 1F</td>
<td>;Output &quot;US&quot; for alpha-</td>
</tr>
<tr>
<td>0387</td>
<td>DA</td>
<td></td>
<td>SEP A</td>
<td>;numeric mode.</td>
</tr>
<tr>
<td>0388</td>
<td>F8</td>
<td>02</td>
<td>LDI 02</td>
<td>;Point RC to the start of</td>
</tr>
<tr>
<td>038A</td>
<td>BC</td>
<td></td>
<td>PHI C</td>
<td>;the first heading stack</td>
</tr>
<tr>
<td>038E</td>
<td>D4</td>
<td>03B6</td>
<td>SUB HEAD</td>
<td>;Output first heading.</td>
</tr>
<tr>
<td>0391</td>
<td>D4</td>
<td>01CE</td>
<td>SUB RDOUT</td>
<td>;Output speed from RD.</td>
</tr>
<tr>
<td>0394</td>
<td>F8</td>
<td>B7</td>
<td>LDI B7</td>
<td>;Point RC to second heading.</td>
</tr>
<tr>
<td>0396</td>
<td>AC</td>
<td></td>
<td>PLO C</td>
<td>;</td>
</tr>
<tr>
<td>0397</td>
<td>D4</td>
<td>03B6</td>
<td>SUB HEAD</td>
<td>;Output second heading.</td>
</tr>
<tr>
<td>039A</td>
<td>9E</td>
<td></td>
<td>CHI E</td>
<td>;Move mark threshold to RF.1</td>
</tr>
<tr>
<td>039B</td>
<td>BF</td>
<td></td>
<td>PHI F</td>
<td>;</td>
</tr>
<tr>
<td>039C</td>
<td>D4</td>
<td>01E2</td>
<td>SUB 2HOUT</td>
<td>;Output it as 2 hex digits.</td>
</tr>
<tr>
<td>039F</td>
<td>F8</td>
<td>CB</td>
<td>LDI CB</td>
<td>;Point RC to third heading.</td>
</tr>
<tr>
<td>03A1</td>
<td>AC</td>
<td></td>
<td>PLO C</td>
<td>;</td>
</tr>
<tr>
<td>03A2</td>
<td>D4</td>
<td>03B6</td>
<td>SUB HEAD</td>
<td>;Output third heading.</td>
</tr>
<tr>
<td>03A5</td>
<td>8E</td>
<td></td>
<td>GLO E</td>
<td>;Put space threshold in RF.1</td>
</tr>
<tr>
<td>03A6</td>
<td>BF</td>
<td></td>
<td>PHI F</td>
<td>;</td>
</tr>
<tr>
<td>03A7</td>
<td>D4</td>
<td>01E2</td>
<td>SUB 2HOUT</td>
<td>;Output it as 2 hex digits.</td>
</tr>
<tr>
<td>03AA</td>
<td>F8</td>
<td>01</td>
<td>LDI 01</td>
<td>;Histogram finished, point</td>
</tr>
<tr>
<td>03AC</td>
<td>AA</td>
<td></td>
<td>PLO A</td>
<td>;RA to routine for port 1,</td>
</tr>
<tr>
<td>03AD</td>
<td>C0</td>
<td>04B8</td>
<td>LBR KCHK</td>
<td>;wait for keyboard input.</td>
</tr>
<tr>
<td>03B0</td>
<td>00</td>
<td></td>
<td>IDL</td>
<td>;IDL's to 03B5 inc.</td>
</tr>
</tbody>
</table>

This short subroutine outputs ASCII headings. Enter with start of data stack in RC, returns when first 00 found.

<table>
<thead>
<tr>
<th>Add.</th>
<th>Code</th>
<th>Label</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>03B6</td>
<td>4C</td>
<td>HEAD</td>
<td>LDA C</td>
</tr>
<tr>
<td>03B7</td>
<td>C5</td>
<td></td>
<td>LSNZ</td>
</tr>
<tr>
<td>03B8</td>
<td>D5</td>
<td></td>
<td>RETURN</td>
</tr>
<tr>
<td>03B9</td>
<td>00</td>
<td></td>
<td>IDL</td>
</tr>
<tr>
<td>03BA</td>
<td>DA</td>
<td></td>
<td>SEP A</td>
</tr>
<tr>
<td>03BB</td>
<td>30</td>
<td>B6</td>
<td>BR HEAD</td>
</tr>
<tr>
<td>03BC</td>
<td>00</td>
<td></td>
<td>IDL</td>
</tr>
</tbody>
</table>
## MORSE PROGRAM

;This is the heading data. Each stack ends with 00.

<table>
<thead>
<tr>
<th>Add.</th>
<th>Code</th>
<th>Character</th>
<th>Add.</th>
<th>Code</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>02B0</td>
<td>53</td>
<td>S</td>
<td>02C9</td>
<td>20</td>
<td>space</td>
</tr>
<tr>
<td>02B1</td>
<td>50</td>
<td>P</td>
<td>02CA</td>
<td>00</td>
<td>end</td>
</tr>
<tr>
<td>02B2</td>
<td>45</td>
<td>E</td>
<td>02CB</td>
<td>20</td>
<td>space</td>
</tr>
<tr>
<td>02B3</td>
<td>45</td>
<td>E</td>
<td>02CC</td>
<td>20</td>
<td>space</td>
</tr>
<tr>
<td>02B4</td>
<td>44</td>
<td>D</td>
<td>02CD</td>
<td>20</td>
<td>space</td>
</tr>
<tr>
<td>02B5</td>
<td>20</td>
<td>space</td>
<td>02CE</td>
<td>20</td>
<td>space</td>
</tr>
<tr>
<td>02B6</td>
<td>00</td>
<td>end</td>
<td>02CF</td>
<td>53</td>
<td>S</td>
</tr>
<tr>
<td>02B7</td>
<td>20</td>
<td>space</td>
<td>02D0</td>
<td>50</td>
<td>P</td>
</tr>
<tr>
<td>02B8</td>
<td>20</td>
<td>space</td>
<td>02D1</td>
<td>41</td>
<td>A</td>
</tr>
<tr>
<td>02B9</td>
<td>20</td>
<td>space</td>
<td>02D2</td>
<td>43</td>
<td>C</td>
</tr>
<tr>
<td>02BA</td>
<td>20</td>
<td>space</td>
<td>02D3</td>
<td>45</td>
<td>E</td>
</tr>
<tr>
<td>02BB</td>
<td>4D</td>
<td>space</td>
<td>02D4</td>
<td>20</td>
<td>space</td>
</tr>
<tr>
<td>02BC</td>
<td>41</td>
<td>A</td>
<td>02D5</td>
<td>54</td>
<td>T</td>
</tr>
<tr>
<td>02BD</td>
<td>52</td>
<td>R</td>
<td>02D6</td>
<td>48</td>
<td>H</td>
</tr>
<tr>
<td>02BE</td>
<td>4B</td>
<td>K</td>
<td>02D7</td>
<td>52</td>
<td>R</td>
</tr>
<tr>
<td>02BF</td>
<td>20</td>
<td>space</td>
<td>02D8</td>
<td>45</td>
<td>E</td>
</tr>
<tr>
<td>02C0</td>
<td>54</td>
<td>T</td>
<td>02D9</td>
<td>53</td>
<td>S</td>
</tr>
<tr>
<td>02C1</td>
<td>48</td>
<td>H</td>
<td>02DA</td>
<td>48</td>
<td>H</td>
</tr>
<tr>
<td>02C2</td>
<td>52</td>
<td>R</td>
<td>02DB</td>
<td>4F</td>
<td>O</td>
</tr>
<tr>
<td>02C3</td>
<td>45</td>
<td>E</td>
<td>02DC</td>
<td>4C</td>
<td>L</td>
</tr>
<tr>
<td>02C4</td>
<td>53</td>
<td>S</td>
<td>02DD</td>
<td>44</td>
<td>D</td>
</tr>
<tr>
<td>02C5</td>
<td>48</td>
<td>H</td>
<td>02DE</td>
<td>20</td>
<td>space</td>
</tr>
<tr>
<td>02C6</td>
<td>4F</td>
<td>O</td>
<td>02DF</td>
<td>00</td>
<td>end</td>
</tr>
<tr>
<td>02C7</td>
<td>4C</td>
<td>L</td>
<td>02C8</td>
<td>44</td>
<td>D</td>
</tr>
</tbody>
</table>
REFERENCES


