DANMARKS INGENIØRAKADEMI
ELEKTROAFDELINGEN
TELETEKNIK
TELEFON (08) 16 05 22
BADEHUSVEJ 1A
9000 AALBORG

0

DGC USER GROUP PROGRAMS.

FFT	
BEAP	
DIS ASSEMBLER	
PIN LIST PROGRAM	
COX I/O PROGRAM	
WIRE LIST PROGRAM	
RESUB	

IEEE TRANSACTIONS ON AUDIO AND ELECTROACOUSTICS, VOL. AU-15, NO. 2, JUNE 1967

What is the Fast Fourier Transform?

G-AE Subcommittee on Measurement Concepts

WILLIAM T. COCHRAN
JAMES W. COOLEY
DAVID L. FAVIN, MEMBER, IEEE
HOWARD D. HELMS, MEMBER, IEEE
REGINALD A. KAENEL, SENIOR MEMBER, IEEE
WILLIAM W. LANG, SENIOR MEMBER, IEEE
GEORGE C. MALING, JR., ASSOCIATE MEMBER, IEEE
DAVID E. NELSON, MEMBER, IEEE,
CHARLES M. RADER, MEMBER, IEEE
PETER D. WELCH

Abstract—**The fast Fourier transform is a computational tool which facilitates signal analysis such as power spectrum analysis and filter simulation by means of digital computers. It is a method for efficiently computing the discrete Fourier transform of a series of data samples (referred to as a time series). In this paper, the discrete Fourier transform of a time series is defined, some of its properties are discussed, the associated fast method (fast Fourier transform) for computing this transform is derived, and some of the computational aspects of the method are presented. Examples are included to demonstrate the concepts involved.**

INTRODUCTION

AN ALGORITHM for the computation of Fourier coefficients which requires much less computational effort than was required in the past was reported by Cooley and Tukey [1] in 1965. This method is now widely known as the "fast Fourier transform," and has produced major changes in computational techniques used in digital spectral analysis, filter simulation, and related fields. The technique has a long and interesting history that has been summarized by Cooley, Lewis, and Welch in this issue [2].

The fast Fourier transform (FFT) is a method for efficiently computing the discrete Fourier transform (DFT) of a time series (discrete data samples). The efficiency of this method is such that solutions to many problems can now be obtained substantially more economically than in the past. This is the reason for the very great current interest in this technique.

The discrete Fourier transform (DFT) is a transform in its own right such as the Fourier integral transform or the Fourier series transform. It is a powerful reversible mapping operation for time series. As the name implies, it has mathematical properties that are entirely analogous to those of the Fourier integral transform. In particular, it defines a spectrum of a time series; multiplication of the transform of two time series corresponds to convolving the time series.

If digital analysis techniques are to be used for analyzing a continuous waveform then it is necessary that the data be sampled (usually at equally spaced intervals of time) in order to produce a time series of discrete samples which can be fed into a digital computer. As is well known [6], such a time series completely represents the continuous waveform, provided this waveform is frequency band-limited and the samples are taken at a rate that is at least twice the highest frequency present in the waveform. When these samples are equally spaced they are known as Nyquist samples. It will be shown that the DFT of such a time series is closely related to the Fourier transform of the continuous waveform from which samples have been taken to form the time series. This makes the DFT particularly useful for power spectrum analysis and filter simulation on digital computers.

The fast Fourier transform (FFT), then, is a highly efficient procedure for computing the DFT of a time series. It takes advantage of the fact that the calculation of the coefficients of the DFT can be carried out iteratively, which results in a considerable savings of computation time. This manipulation is not intuitively obvious, perhaps explaining why this approach was overlooked for such a long time. Specifically, if the time series consists of $N=2^n$ samples, then about $2nN=2N\cdot\log_2 N$ arithmetic operations will be shown to be required to evaluate all N associated DFT coefficients. In comparison with the number of operations required for the calculation of the DFT coefficients with straightforward procedures (N^2), this number is so small when N is large as to completely change the computationally economical approach to various problems. For example, it has been reported that for $N=8192$ samples, the computations require about five seconds

Manuscript received March 10, 1967.

W. T. Cochran, D. L. Favin, and R. A. Kaenel are with Bell Telephone Laboratories, Inc., Murray Hill, N. J.

H. D. Helms is with Bell Telephone Laboratories, Inc., Whippany, N. J.

J. W. Cooley and P. D. Welch are with the IBM Research Center, Yorktown Heights, N. Y.

W. W. Lang and G. C. Maling are with the IBM Corporation, Poughkeepsie, N. Y.

C. M. Rader is with Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Mass. (Operated with support from the U. S. Air Force.)

D. E. Nelson is with the Electronics Division of the General Dynamics Corporation, Rochester, N. Y.

for the evaluation of all 8192 DFT coefficients on an IBM 7094 computer. Conventional procedures take on the order of half an hour.

The known applications where a substantial reduction in computation time has been achieved include: 1) computation of the power spectra and autocorrelation functions of sampled data [4]; 2) simulation of filters [5]; 3) pattern recognition by using a two-dimensional form of the DFT; 4) computation of bispectra, cross-covariance functions, cepstra and related functions; and 5) decomposing of convolved functions.

The Discrete Fourier Transform (DFT)

Definition of the DFT and its Inverse

Since the FFT is an efficient method for computing the DFT it is appropriate to begin by discussing the DFT and some of the properties that make it so useful a transformation. The DFT is defined by[1]

$$A_r = \sum_{k=0}^{N-1} X_k \exp(-2\pi jrk/N) \quad r = 0, \cdots, N-1 \quad (1)$$

where A_r is the rth coefficient of the DFT and X_k denotes the kth sample of the time series which consists of N samples and $j=\sqrt{-1}$. The X_k's can be complex numbers and the A_r's are almost always complex. For notational convenience (1) is often written as

$$A_r = \sum_{k=0}^{N-1} (X_k) W^{rk} \qquad r = 0, \cdots, N-1 \qquad (2)$$

where

$$W = \exp(-2\pi j/N). \qquad (3)$$

Since the X_k's are often values of a function at discrete time points, the index r is sometimes called the "frequency" of the DFT. The DFT has also been called the "discrete Fourier transform" or the "discrete time, finite range Fourier transform."

There exists the usual inverse of the DFT and, because the form is very similar to that of the DFT, the FFT may be used to compute it.

The inverse of (2) is

$$X_l = (1/N) \sum_{r=0}^{N-1} A_r W^{-rl} \quad l = 0, 1, \cdots, N-1. \quad (4)$$

This relationship is called the inverse discrete Fourier transform (IDFT). It is easy to show that this inversion is valid by inserting (2) into (4)

$$X_l = \sum_{r=0}^{N-1} \sum_{k=0}^{N-1} (X_k/N) W^{r(k-l)}. \qquad (5)$$

Interchanging in (5) the order of summing over the indices r and k, and using the orthogonality relation

$$\sum_{r=0}^{N-1} \exp(2\pi j(n-m)r/N) = N, \text{ if } n \equiv m \bmod N$$
$$= 0, \text{ otherwise} \qquad (6)$$

establishes that the right side of (5) is in fact equal to X_k.

It is useful to extend the range of definition of A_r to all integers (positive and negative). Within this definition it follows that

$$A_r = A_{N+r} = A_{2N+r} = \cdots \qquad (7)$$

Similarly,

$$X_l = X_{N+l} = X_{2N+l} = \cdots. \qquad (8)$$

Relationships between the DFT and the Fourier Transform of a Continuous Waveform

An important property that makes the DFT so eminently useful is the relationship between the DFT of a sequence of Nyquist samples and the Fourier transform of a continuous waveform, that is represented by the Nyquist samples. To recognize this relationship, consider a frequency band-limited waveform $g(t)$ whose Nyquist samples, X_k, vanish outside the time interval $0 \le t \le NT$

$$g(t) = \sum_{k=0}^{N-1} \frac{\sin(\pi(t-kT)/T)}{(\pi(t-kT)/T)} \cdot X_k \qquad (9)$$

where T is the time spacing between the samples. A periodic repetition of $g(t)$ can be constructed that has identically the same Nyquist samples in the time interval $0 \le t \le NT$

$$g_p(t) = \sum_{l} \sum_{k=0}^{N-1} X_k \cdot \frac{\sin(\pi(t-kT-lNT)/T)}{(\pi(t-kT-lNT)/T)}. \qquad (10)$$

Let the Fourier transform of $g(t)$ be $G(f)$. As is well known [6], this transform is exactly specified at discrete frequencies by the complex Fourier series coefficients of $g_p(t)$. From this it follows:

$$\frac{G(n/NT)}{NT} = D_n$$
$$= (1/NT) \int_0^{NT} g_p(t) \cdot \exp(-2\pi jnt/NT) \cdot dt$$
$$= (1/NT) \sum_{k=0}^{N-1} X_k \cdot \exp(-2\pi jnkT/NT) \qquad (11)$$

where $|n| \le N/2$ due to the spectral bandwidth limitation implicitly assumed by the sampling theorem underlying the validity of Nyquist samples.

Comparing (11) and (1) it is seen that they are exactly the same except for a factor of NT and (r, n) are both unbounded. That is,

$$N \cdot A_r = D_n \text{ for } r = n \text{ and } T = 1 \text{ second.} \qquad (12)$$

The bounds specified for r and n require a correspondence which depends on (7)

[1] The definition of the DFT is not uniform in the literature. Some authors use A_r/N as the DFT coefficients, others use $A_r/\sqrt{N}$, still others use a positive exponent.

$$\frac{G(n/NT)}{NT} = D_n = N \cdot A_r$$

where

$$n = r \qquad \text{for } n = 0, 1, \cdots, q < N/2,$$

and

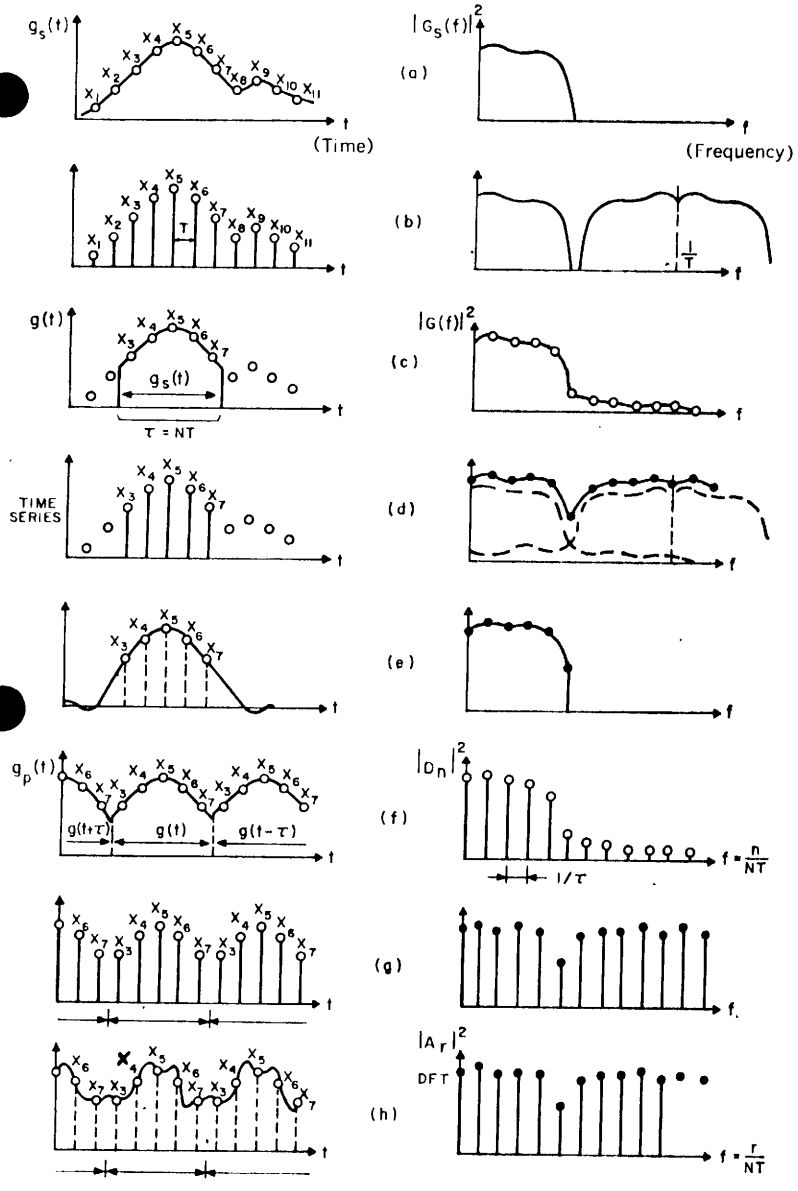
$$n = N - r \quad \text{for } n = -1, -2, \cdots, -q > -N/2 \quad (13)$$

and

$$\frac{G(n/NT)}{NT} = D_n = N \cdot A_r/2 \qquad \text{for } n = N/2. \quad (14)$$

Equations (13) and (14) give a direct relationship between the DFT coefficients and the Fourier transform at discrete frequencies for the waveform stipulated by (9). A one-to-one correspondence could have been obtained if the running variable r had been bounded by $\pm N/2$. This, however, would have required distinguishing between even and odd values of N, a distinction avoided by keeping r positive.

A waveform of the type considered by (9) is shown in Fig. 1(e). It is usually obtained as an approximation of a frequency band-limited source waveform [such as the one sketched in Fig. 1(a)] by truncating the Nyquist sample series of this waveform, and reconstructing the continuous waveform corresponding to the truncated Nyquist sample series [Fig. 1(b), (d), and (e)]. Notwithstanding the identity of the Nyquist samples of this reconstructed waveform and the frequency band-limited source waveform, these waveforms differ in the truncation interval [Fig. 1(c) and (e)]. The difference is usually referred to as aliasing distortion; the mechanics of this distortion is most apparent in the frequency domain [Fig. 1(c)–(e)]. It can be made negligibly small by choosing a sufficiently large product of the frequency bandwidth of the source waveform and the duration of the truncation interval [6] (e.g., N is greater than ten).



(a) Frequency-band-limited source waveform.
(b) Nyquist samples of the frequency band-limited source waveform.
(c) Truncated source waveform.
(d) Truncated series of Nyquist samples of the source waveform.
(e) Frequency-band-limited waveform whose Nyquist samples are identical to the truncated series of Nyquist samples of the source waveform.
(f) Periodic continuation of the truncated source waveform.
(g) Periodic continuation of the truncated series of Nyquist samples of the source waveform.
(h) DFT coefficients interpreted as Fourier series coefficients producing complex waveform.

Fig. 1. Related waveforms and their corresponding spectra as defined by the Fourier transforms (integral transforms for energy-limited waveforms; series transform for periodic waveforms).

These aliasing distortions are carried over directly to the discrete spectra of the periodically repeated waveforms [Fig. 1(f) and (g)], and appear correspondingly in the DFT of the truncated series of Nyquist samples [Fig. 1(h)]. It may be of interest to observe that the waveform corresponding to the DFT coefficients interpreted as Fourier series coefficients is complex [Fig. 1(h)].

Some Useful Properties of the DFT

Another property that makes the DFT eminently useful is the convolution relationship. That is, the IDFT of the product of two DFTs is the periodic mean convolution of the two time series of the DFTs. This relationship proves very useful when computing the filter output as a result of an input waveform; it becomes especially effective when computed by the FFT. A derivation of this property is given in Appendix A.

Other properties of the DFT are in agreement with the corresponding properties of the Fourier integral transform, perhaps with slight modifications. For example, the DFT of a time series circularly shifted by h is the DFT of the time series multiplied by W^{-rh}. Furthermore, the DFT of the sum of two functions is the sum of the DFT of the two functions. These properties are readily derived using the definition of the DFT. These and other properties have been compiled by Gentleman and Sande [7].

The Fast Fourier Transform

General Description of the FFT

As mentioned in the Introduction, the FFT is an algorithm that makes possible the computation of the DFT of a time series more rapidly than do other algorithms available. The possibility of computing the DFT by such a fast algorithm makes the DFT technique important. A comparison of the computational savings that may be achieved through use of the FFT is summarized in Table I for various computations that are frequently performed. It is important to add that the computational efforts listed represent comparable upper bounds; the actual efforts depend on the number N and the programming ingenuity applied [7].

It may be useful to point out that the FFT not only reduces the computation time; it also substantially reduces round-off errors associated with these computations. In fact, both computation time and round-off error essentially are reduced by a factor of $(\log_2 N)/N$ where N is the number of data samples in the time series. For example, if $N = 1024 = 2^{10}$, then $N \cdot \log_2 N = 10\,240$ [7], [9]. Conventional methods for computing (1) for $N = 1024$ would require an effort proportional to $N^2 = 1\,048\,576$, more than 50 times that required with the FFT.

The FFT is a clever computational technique of sequentially combining progressively larger weighted sums of data samples so as to produce the DFT coefficients as defined by (2). The technique can be interpreted in terms of combining the DFTs of the individual data samples such that the occurrence times of these samples are taken into account sequentially and applied to the DFTs of progressively larger mutually exclusive subgroups of data samples, which are combined to ultimately produce the DFT of the complete series of data samples. The explanation of the FFT algorithm adopted in this paper is believed to be particularly descriptive for programming purposes.

TABLE I

Comparison of the Number of Multiplications Required Using "Direct" and FFT Methods

Operation	Formula	Approximate Number of Multiplications (upper comparable bounds)	
		Direct	FFT
Discrete Fourier Transform (DFT)	$\sum_{k=0}^{N-1} X_k e^{2\pi jrk/N} \quad r = 1, 2, \cdots, N-1$	N^2	$2N \log_2 N$
Filtering (Convolution)	$\sum_{k=0}^{N-1} X_k Y_{n-k} \quad n = 0, 1, \cdots, N-1$	N^2	$3N \log_2 N$
Autocorrelation Functions	$\sum_{k=0}^{N-1-r} X_k X_{r+k} \quad r = 0, 1, \cdots, N-1$	$\frac{N}{4}\left(\frac{N}{2}+3\right)$	$3N \log_2 N$
Two-Dimensional Fourier Transform (Pattern Analysis)	$\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} X_{k,l} e^{-2\pi j(kq+rl/N)} \quad r, q = 0, 1, \cdots, N-1$	N^4	$4N^2 \log_2 N$
Two-Dimensional Filtering	$\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} X_{k,l} Y_{q-k,r-l} \quad q, r = 1, 2, \cdots, N-1$	N^4	$3N^2 \log_2 N$

Conventional Forms of the FFT

Decimation in Time: The DFT [as per (2)] and its inverse [see (4)] are of the same form so that a procedure, machine, or sub-routine capable of computing one can be used for computing the other by simply exchanging the roles of X_k and A_r, and making appropriate scale-factor and sign changes. The two basic forms of the FFT, each with its several modifications, are therefore equivalent. However, it is worth distinguishing between them and discussing them separately. Let us first consider the form used by Cooley and Tukey [1] which shall be called *decimation in time*. Reversing the roles of A_r and X_k gives the form called *decimation in frequency*, which will be considered afterwards.

Suppose a time series having N samples [such as X_k shown in Fig. 2(a)] is divided into two functions, Y_k and Z_l, each of which has only half as many points ($N/2$). The function Y_k is composed of the even-numbered points ($X_0, X_2, X_4 \cdots$), and Z_k is composed of the odd numbered points ($X_1, X_3, X_5 \cdots$). These functions are shown in Fig. 2(b) and (c), and we may write them formally as

$$\begin{aligned} Y_k &= X_{2k} \\ Z_k &= X_{2k+1} \end{aligned} \qquad k = 0, 1, 2, \cdots, \frac{N}{2} - 1. \tag{15}$$

Since Y_k and Z_k are sequences of $N/2$ points each, they have discrete Fourier transforms defined by

$$\begin{aligned} B_r &= \sum_{k=0}^{(N/2)-1} Y_k \exp(-4\pi jrk/N) \\ C_r &= \sum_{k=0}^{(N/2)-1} Z_k \exp(-4\pi jrk/N) \end{aligned} \qquad r = 0, 1, 2, \cdots, \frac{N}{2} - 1. \tag{16}$$

The discrete Fourier transform that we want is A_r, which we can write in terms of the odd- and even-numbered points

$$A_r = \sum_{k=0}^{(N/2)-1} \left\{ Y_k \exp(-4\pi jrk/N) + Z_k \exp\left(-\frac{2\pi jr}{N}[2k+1]\right) \right\} \qquad r = 0, 1, 2, \cdots, N-1 \tag{17}$$

or

$$A_r = \sum_{k=0}^{(N/2)-1} Y_k \exp(-4\pi jrk/N) + \exp(-2\pi jr/N) \sum_{k=0}^{(N/2)-1} Z_k \exp(-4\pi jrk/N) \tag{18}$$

which, using (16), may be written in the following form:

$$A_r = B_r + \exp(-2\pi jr/N)C, \quad 0 \le r < N/2. \tag{19}$$

For values of r greater than $N/2$, the discrete Fourier transforms B_r and C_r repeat periodically the values taken on when $r < N/2$. Therefore, substituting $r + N/2$ for r in (19), we obtain

$$\begin{aligned} A_{r+N/2} &= B_r + \exp\left(-2\pi j\left[r + \frac{N}{2}\right]/N\right) C_r \qquad 0 \le r < N/2 \\ &= B_r - \exp(-2\pi jr/N)C_r \qquad 0 \le r < N/2. \end{aligned} \tag{20}$$

By using (3), (19) and (20) may be written as

$$A_r = B_r + W^r C_r \qquad 0 \le r < N/2 \tag{21}$$

$$A_{r+N/2} = B_r - W^r C_r \qquad 0 \le r < N/2. \tag{22}$$

From (21) and (22), the first $N/2$ and last $N/2$ points of the discrete Fourier transform of X_k (a sequence having N samples) can be simply obtained from the DFT of Y_k and Z_k, both sequences of $N/2$ samples.

Assuming that we have a method which computes discrete Fourier transforms in a time proportional to the square of the number of samples, we can use this algorithm to compute the transforms of Y_k and Z_k, requiring a time proportional to $2(N/2)^2$, and use (21) and (22) to find A_r with additional N operations. This is illustrated in the signal flow graph of Fig. 3. The points on the left are the values of X_k (i.e., Y_k and Z_k), and the points on the right are the points of the discrete Fourier transform, A_r. For simplicity, Fig. 3 is drawn for the case where X_k is an eight-point function, and advantage is taken of the fact that $W^n = -W^{n-N/2}$, as per (3).

However, since Y_k and Z_k are to be transformed, and since we have shown that the computation of the DFT of N samples can be reduced to computing the DFTs of two sequences of $N/2$ samples each, the computation of B_k (or C_k) can be reduced to the computation of sequences of $N/4$ samples. These reductions can be carried out as long as each function has a number of samples that is divisible by 2. Thus, if $N = 2^n$ we can make n such reductions, applying (15), (21), and (22) first for N, then for $N/2$, $\cdots$, and finally for a two-point function. The discrete Fourier transform of a one-point function is, of course, the sample itself. The successive reduction of an eight-point discrete Fourier transform, begun in Fig. 3, is continued in Figs. 4 and 5. In Fig. 5 the operation has been completely reduced to complex multiplications and additions. From the signal flow graph there are 8 by 3 terminal nodes and 2 by 8 by 3 arrows, corresponding to 24 additions and 48 multiplications. Half of the multiplications can be omitted since the transmission indicated by the arrow is unity. Half of the remaining multiplications are also easily eliminated, as we shall see below. Thus, in general, $N \cdot \log_2 N$ complex additions and, at most, $\frac{1}{2}N \cdot \log_2 N$ complex multi-

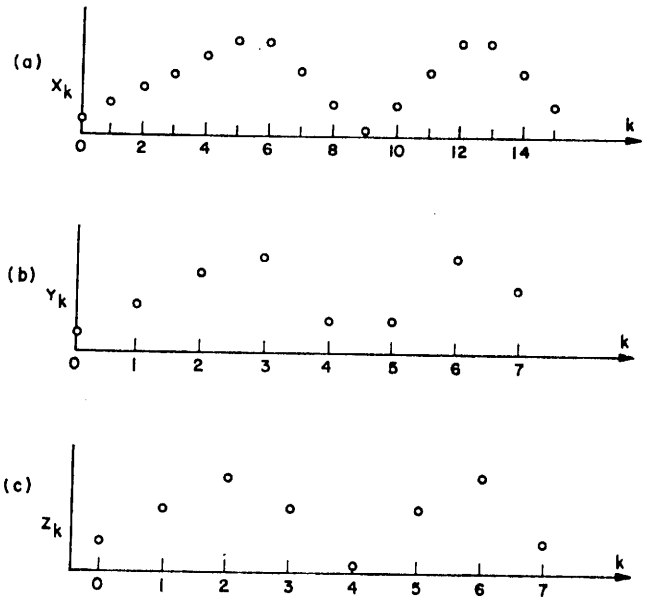


Fig. 2. Decomposition of a time series into two part-time series, each of which consists of half the samples.

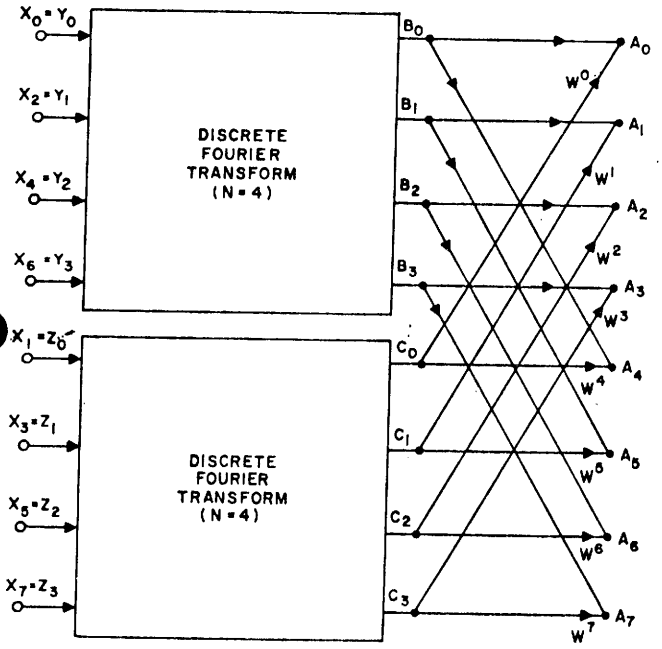


Fig. 3. Signal flow graph illustrating the reduction of endpoint DFT to two DFTs of $N/2$ points each, using decimation in time. The signal flow graph may be unfamiliar to some readers. Basically it is composed of dots (or nodes) and arrows (transmissions). Each node represents a variable, and the arrows terminating at that node originate at the nodes whose variables contribute to the value of the variable at that node. The contributions are additive, and the weight of each contribution, if other than unity, is indicated by the constant written close to the arrowhead of the transmission. Thus, in this example, the quantity A_7 at the bottom right node is equal to $B_3 + W_7 \times C_3$. Operations other than addition and constant multiplication must be clearly indicated by symbols other than · or ——→.

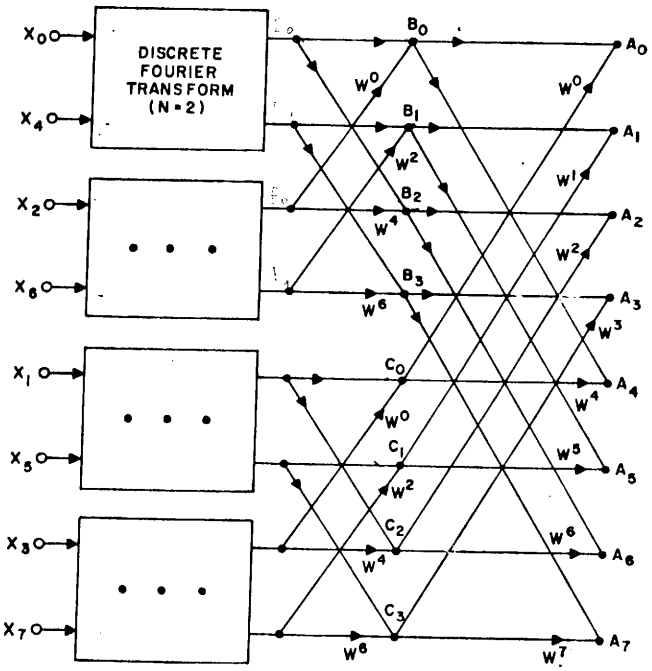


Fig. 4. Signal flow graph illustrating further reduction of the DFT computation suggested by Fig. 3.

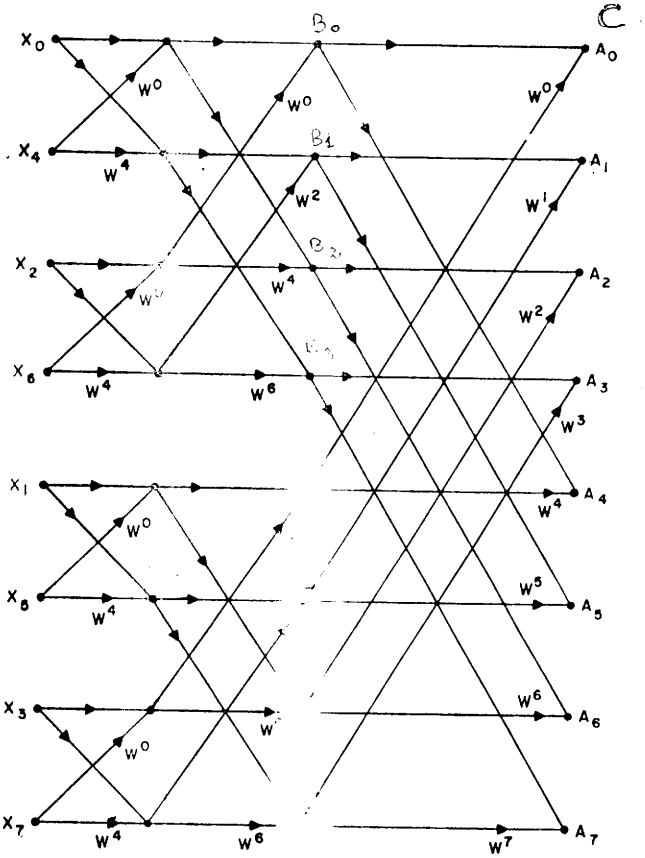


Fig. 5. Signal flow graph illustrating the computation of the DFT when the operations involved are completely reduced to multiplications and additions.

plications are required for computation of the discrete Fourier transform of an N point sequence, where N is a power of 2.

When N is not a power of 2, but has a factor p, the development of equations analogous to (15) through (22) is possible by forming p different sequences, $Y_j^{(i)} = X_{pk+i}$, each having N/p samples. Each of these sequences has a DFT $B_r^{(i)}$, and the DFT of the sequence X_k can be computed from the p simpler DFTs with pN complex multiplications and additions. That is,

$$A_{r+m(N/p)} = \sum_{i=0}^{p-1} B_r^{(i)} W^{i[r+m(N/p)]}$$

$$m = 0, 1, 2, \cdots, p-1$$

$$r = 0, 1, 2, \cdots, \frac{N}{p} - 1. \quad (23)$$

The computation of the DFTs can be further simplified if N has additional prime factors.

Further information about the fast Fourier transform can be extracted from Fig. 5. For example, if the input sequence X_k is stored in computer memory in the order

$$X_0, X_4, X_2, X_6, X_1, X_5, X_3, X_7, \quad (24)$$

as in Fig. 5, the computation of the discrete Fourier transform may be done "in place," that is, by writing all intermediate results over the original data sequence, and writing the final answer over the intermediate results. Thus, no storage is needed beyond that required for the original N complex numbers. To see this, suppose that each node corresponds to two memory registers (the quantities to be stored are complex). The eight nodes farthest to the left in Fig. 5 then represent the registers containing the shuffled order input data. The first step in the computation is to compute the contents of the registers represented by the eight nodes just to the right of the input nodes. But each pair of input nodes affects only the corresponding pair of nodes immediately to the right, and if the computation deals with two nodes at a time, the newly computed quantities may be written into the registers from which the input values were taken, since the input values are no longer needed for further computation. The second step, computation of the quantities associated with the next vertical array of nodes to the right, also involves pairs of nodes although these pairs are now two locations apart instead of one. This fact does not change the property of "in place" computation, since each pair of nodes affects only the pair of nodes immediately to the right. After a new pair of results is computed, it may be stored in the registers which held the old results that are no longer needed. In the computation for the final array of nodes, corresponding to the values of the DFT, the computation involves pairs of nodes separated by four locations, but the "in place" property still holds.

For this version of the algorithm, the initial shuffling of the data sequence, X_k, was necessary for the "in place" computation. This shuffling is due to the repeated movement of odd-numbered members of a sequence to the end of the sequence during each stage of the reduction, as shown in Figs. 3, 4, and 5. This shuffling has been called *bit reversal*[2] because the samples are stored in bit-reversed order; i.e., $X_4 = X_{(100)_2}$ is stored in position $(001)_2 = 1$, etc. Note that the initial data shuffling can also be done "in place."

Variations of Decimation in Time: If one so desires, the signal flow graph shown in Fig. 5 can be manipulated to yield different forms of the *decimation in time* version of the algorithm. If one imagines that in Fig. 5 all the nodes on the same horizontal level as A_1 are interchanged with all the nodes on the same horizontal level as A_4, and all the nodes on the level of A_3 are interchanged with the nodes on the level of A_6, *with the arrows carried along with the nodes*, then one obtains a flow graph like that of Fig. 6.

For this rearrangement one need not shuffle the original data into the bit-reversed order, but the resulting spectrum needs to be *unshuffled*. An additional disadvantage might be that the powers of W needed in the computation are in bit-reversed order. Cooley's original description of the algorithm [1] corresponds to the flow graph of Fig. 6.

A somewhat more complicated rearrangement of Fig. 5 yields the signal flow graph of Fig. 7. For this case both the input data and the resulting spectrum are in "natural" order, and the coefficients in the computation are also used in a natural order. However, the computation may no longer be done "in place." Therefore, at least one other array of registers must be provided. This signal flow graph, and a procedure corresponding to it, are due to Stockham [8].

Decimation in Frequency: Let us now consider a second, quite distinct, form of the fast Fourier transform algorithm, *decimation in frequency*. This form was found independently by Sande [7], and Cooley and Stockham [8]. Let the time series X_k have a DFT A_r. The series and the DFT both contain N terms. As before, we divide X_k into two sequences having $N/2$ points each. However, the first sequence, Y_k, is now composed of the first $N/2$ points in X_k, and the second, Z_k, is composed of the last $N/2$ points in X_k. Formally, then

$$Y_k = X_k$$

$$k = 0, 1, 2, \cdots, \frac{N}{2} - 1. \quad (25)$$

$$Z_k = X_{k+N/2}$$

[2] This is a special case of digit reversal where the radix of the address is 2; more general digit reversals are available for transforms with other radices.

A Useful Computational Variation: It may be worth pointing out here how some programming simplicity is realized when the factors p and $q = N/p$ are relatively prime. As described by Cooley, Lewis, and Welch, [2], the "twiddle factor" W^{ir} of (23) can be eliminated by choosing subsequences of the X_k's that are different than those used before. The DFT computations are then conveniently performed in two stages.

1) Compute the q-point transforms

$$B_r^{(i)} = \sum_{k=0}^{q-1} Y_k^{(i)} \cdot W^{pkr} \qquad \begin{matrix} i = 0, 1, \cdots, p-1 \\ r = 0, 1, \cdots, q-1 \end{matrix} \tag{31}$$

of each of the p sequences

$$Y_k^{(i)} = X_{pk+qi} \qquad \begin{matrix} i = 0, 1, \cdots, p-1 \\ k = 0, 1, \cdots, q-1. \end{matrix} \tag{32}$$

2) Compute, then, the p-point transforms

$$A_s = \sum_{i=0}^{p-1} B_r^{(i)} \cdot W^{qim} \tag{33}$$

of the q sequences $B_r^{(i)}$, where

$$s = r \cdot p(p)_q^{-1} + m \cdot q(q)_p^{-1} \qquad (\text{mod } N, 0 \leq s < N) \tag{34}$$

and the notation $(p)_q^{-1}$ is meant to represent the reciprocal of p, mod q, i.e., the solution of $p(p)_q^{-1} > 1 \pmod q$.

Conclusion

The integral transform method has been one of the foundations of analysis for many years because of the ease with which the transformed expressions may be manipulated, particularly in such diverse areas as acoustic wave propagation, speech transmission, linear network theory, transport phenomena, optics, and electromagnetic theory. Many problems which are particularly amenable to solution by integral transform methods have not been attacked by this method in the past because of the high cost of obtaining numerical results this way.

The fast Fourier transform has certainly modified the economics of solution by transform methods. Some new applications are presented in this special issue, and further interesting and profitable applications probably will be found during the next few years.

Appendix

As is well known, if the filter impulse response is frequency-band-limited to $1/2T$ Hz and is given by its Nyquist samples Y_k spaced T second apart, and furthermore, if the input waveform is also frequency-band-limited to $1/2T$ Hz and given by its Nyquist samples X_k spaced T second apart, then the filter output waveform is also frequency-band-limited to $1/2T$ Hz and completely specified by its Nyquist samples Z_s spaced T second apart

$$Z_s = \sum_{k=0}^{s} X_k \cdot Y_{s-k} = \sum_{l=0}^{s} X_{s-l} \cdot Y_l. \tag{35}$$

The convolution relationship facilitates computation of this equation.

To prove the convolution relationship, let the DFT of the X_k's be A_r and correspondingly the DFT of the Y_k's be B_r. The IDFT of the product of $A_r \cdot B_r$ then becomes [see (4)]

$$\begin{aligned} \left(\frac{1}{N^2}\right) \sum_{r=0}^{N-1} A_r B_r W^{-rs} &= \frac{1}{N^2} \sum_{r=0}^{N-1} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} X_k Y_l W^{r(k+l-s)} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} X_k Y_l \sum_{r=0}^{N-1} \frac{W^{r(k+l-s)}}{N} \\ &= \frac{1}{N} \sum_{k=0}^{s} X_k Y_{s-k} + \frac{1}{N} \sum_{k=s+1}^{N-1} X_k Y_{N+s-k} \\ &= \left(\frac{1}{N}\right) \cdot Z_s + \text{perturbation term.} \end{aligned} \tag{36}$$

If the first $N/2$ samples of each of the two time series (X_k) and (Y_k) are assumed to be identically zero, then the perturbation term of (36) is zero so that the IDFT of the product of the two DFTs multiplied by N is equal to the convolution product Z_s of (35). Since it is always possible to select the time series to be convolved such that half of the samples are zero, the convolution relationship for the DFT can be used to compute the convolution product [see (35)] of two time series.

It is useful to point out that if $A_r = B_r$, a periodic autocorrelation function emerges.

References

[1] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. of Comput.*, vol. 19, pp. 297–301, April 1965.

[2] J. W. Cooley, P. A. W. Lewis, and P. D. Welch, "Historical notes on the fast Fourier transform" this issue, p. 76–79.

[3] R. B. Blackman and J. W. Tukey, *The Measurement of Power Spectra*. New York: Dover, 1959.

[4] C. Bingham, M. D. Godfrey, and J. W. Tukey, "Modern techniques of power spectrum estimation," this issue, p. 56–66.

[5] T. G. Stockham, "High speed convolution and correlation," *1966 Spring Joint Computer Conf., AFIPS Proc.*, vol. 28. Washington, D. C.: Spartan, 1966, pp. 229–233.

[6] W. T. Cochran, J. J. Downing, D. L. Davin, H. D. Helms, R. A. Kaenel, W. W. Lang, and D. E. Nelson, "Burst measurements in the frequency domain," *Proc. IEEE*, vol. 54, pp. 830–841, June 1966.

[7] W. M. Gentleman and G. Sande, "Fast Fourier transforms—for fun and profit," *1966 Fall Joint Computer Conf. AFIPS Proc.*, vol. 29. Washington, D. C.: Spartan, 1966, pp. 563–578.

[8] Private communication.

[9] J. W. Cooley, "Applications of the fast Fourier transform method," *Proc. of the IBM Scientific Computing Symp.*, June 1966.

FAST FOURIER TRANSFORM AND INVERSE.

Programmer - Jon Sluthalt
Scripps Institution of Oceanography
Date Submitted: 15 April 1971

The discrete transform (DFT) of a complex series F(N*T) is

$$G(K*Q) = \frac{1}{NUM} \sum_{N=0}^{NUM-1} F(N*T)*\exp(-J*Q*T*N*K) \text{ where } Q=2\pi/(NUM*T),$$

$J = \sqrt{-1}$.

The inverse DFT is

$$F(L*T) = \sum_{K=0}^{NUM-1} G((K*Q))*\exp(J*Q*T*K*L).$$

The factor (1/NUM) is not included in this program.

Definitions:

NUM is the number of sample points. It must be an integer power of 2.

M is the integer power of 2. Thus NUM = 2**M.

Method:

Decimation in time, computation in place, integer arithmetic, overflow test by method 3 described by Welch, 1969, YEEE AU-17, p. 152.

Calling Sequence:

JSR @ 103 ; 103 contains 5062 this listing.

<FLAG> ; FLAG = 0 for forward transform, = | for inverse.

<NUM> ; Array length (LEQ 2000 octal, note 3).

<M> ; $M = \log_2(NUM)$.

<ALPHA> ; Array storage address (note 1).

<SCNT> ; Scaling constant (note 2).

<RETURN>;

Notes:

1. Array order: Input and transformed output are stored in following order starting at location ALPHA: RI (0), RI (1),...,RI(NUM-1), Im (0),...,Im (NUM-1).

2. Scaling: During an array calculation the program tests and "scales" (divides by two) the array if necessary to prevent overflow. The output must be multiplied by 2**(SCNT-M) for the forward transform and 2**(SCNT) for the inverse.

3. Maximum length: The input series is limited to 1024 (decimal) complex values. A real series of double length can be transformed for broad band spectral purposes by treating it as a complex series of half length.

4. Cosine array: locations 4400 through 5000 contain all the cosines necessary to compute the sinusoidal factors of exp ($2\pi j$*N*K/NUM). They are in a format whereby 1.0 is represented by 40000(octal).

Timing:

Requires approximately 15 seconds on NOVA (no hardware multiply).

FAST FOURIER TRANSFORM FOR THE NOVA

1) Our program takes a time series, treats it as several series of one point each, and then combines their transforms to produce the transform of the original series. Let the Discrete Fourier Transform (DFT) of a discrete time series, F(NT), be $F(k\Omega)$ where $\Omega = \frac{2\pi}{(NUM)T}$ is the frequency and T is the sampling interval time. All values in this report are assumed to be complex integers. By definition,

$$F(k\Omega) = \frac{1}{(NUM)} \sum_{N=0}^{(NUM)-1} f(NT) \exp(-i\Omega kNT) \qquad (1)$$

is the forward transform and

$$f(\ell T) = \sum_{k=0}^{(NUM)-1} F(k\Omega) \exp(+i\Omega \ell kT) \qquad (2)$$

is the inverse DFT. N, k, and ℓ run from zero to NUM-1. The inverse DFT can be computed by changing a parameter in the calling sequence of the program. The factor $\frac{1}{(NUM)}$ is not included in the calculation by the program.

2) The Fast Fourier Transform, FFT, is an algorithm for calculating the DFT. There are two basic forms of the FFT; Decimation in Time and Decimation in Frequency, see Gold and Rader, Chapter 6. Our program is based on Decimation in Time. The number of sample points must be equal to 2^M where M is an integer less than or equal to 10 (decimal). The essence of the Decimation in Time method follows: (For notational ease f(NT) will be denoted as f_N, etc.).

Suppose we have a series f_N of (NUM) points with its transform F. Let us break up f_N into two shorter series g_ℓ and h_ℓ with $\ell = 0, \ldots, \frac{(NUM)}{2} - 1$. In fact, we make g_ℓ to be the even points of f_N and h_ℓ the odds. We have then, $f_{2\ell} = g_\ell$ and $f_{2\ell+1} = h_\ell$. Designating their transforms by capitals, we have that the DFT's of g_ℓ and h_ℓ are,

$$G_k = \sum_{\ell=0}^{\frac{NUM}{2}-1} g_\ell \, (W^2)^{\ell k}, \tag{3 a}$$

and

$$H_k = \sum_{\ell=0}^{NUM/2-1} h_\ell \, (W^2)^{\ell k}, \tag{3 b}$$

where $W = \exp(-i \Omega T)$.

We can combine G_k and H_k to get F_k, the desired transform,

$$F_k = \sum_{\ell=0}^{\frac{NUM}{2}-1} (g_\ell W^{2\ell k} + h_\ell W^{(2\ell+1)k}) = G_k + W^k H_k) \quad k = 0, \ldots, NUM-1. \tag{4}$$

Clearly G_k and H_k are periodic in NUM/2 points; therefore, (4) becomes

$$F_k = \begin{cases} G_k + W^k H_k) & 0 \leq k < \frac{NUM}{2}, \\ G_{k-\frac{NUM}{2}} + W^k H_{k-\frac{NUM}{2}}, & \frac{NUM}{2} \leq k < NUM. \end{cases} \tag{5}$$

We can continue this process of decomposing the series by choosing the even and odd points of both g_ℓ and h_ℓ; i.e., if we had

$$a_m = g_{2m} = f_{4m}, \qquad c_m = h_{2m} = f_{4m+1}$$

and

$$b_m = g_{2m+1} = f_{4m+2} \qquad d_m = h_{2m+1} = f_{4m+3},$$

then F_k would be broken up into the sum of four transforms, A_k, B_k, C_k and D_k weighted by exponentials and having a period of NUM/4.

Since NUM = 2^M then we can break down f_N until it is a reordered sequence of NUM series each with one element. Since the DFT of a single point is the value of that point we have reconstituted f_N to get F_k by simple additions and multiplications. This requires a number of arithmetic operations proportional to (NUM) x $\log_2$ (NUM).

3) The program uses an iterative method derived from the flow chart shown in figure 1. Each vertical array (collection of nodal points) represents a collection of partial sums. The actual program is chronologically the reverse of what was stated in paragraph (2), i.e., the program first treats the input series as NUM separate series. It takes the transform of these and gets NUM one point transforms. The program then combines these transforms in a fashion analogous to the decomposition described to get the desired transform, F_N . The evaluation of the sinusoidal factor shown in Figure 1 is simplified by recognizing that $W^{g(N)} = -W^{g(N+NUM/2^I)}$*. This is true since each sub-sequence can be seen as discrete points on a circle where 2π radians equals the period of the sub-sequence: therefore, $\frac{NUM}{2^I}$ represents an addition of π to the angle identified with N or, in other words, a negation of $W^{g(N)}$.

4) During arithmetic operations it is necessary to test for overflow.

* g(N) = N with the last (M=I) binary digits representing N reversed. The $\frac{NUM}{2^I}$ factor is called the "Node Run" in the annotation of our program.

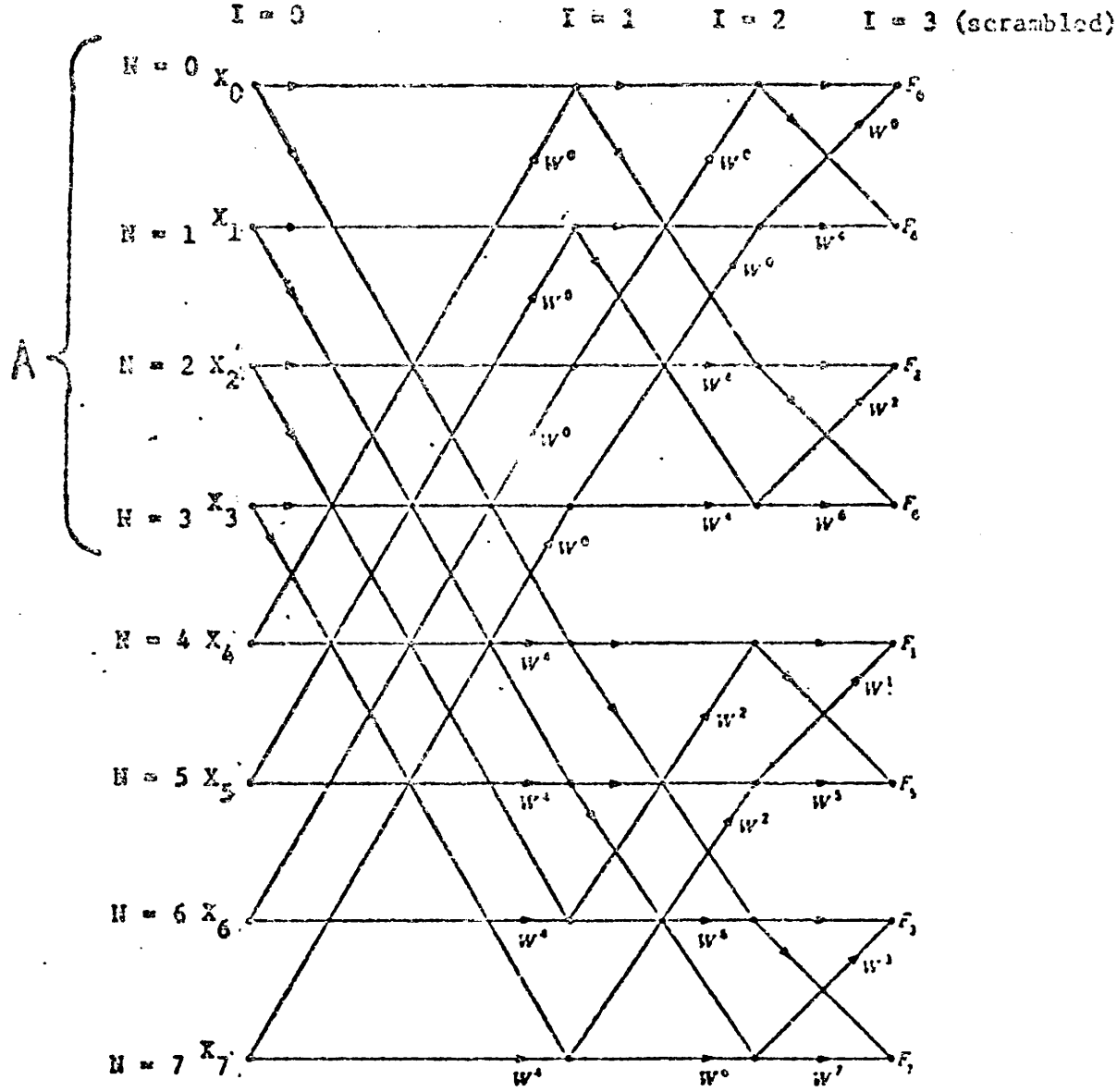


Figure 1. Example for NUM=2^3, after Gold & Rader. The lines in the flow chart indicate the values to be added to determine the value of the node point in the Ith array. If there is a factor printed along the arrow of the line (e.g. W^0), it is multiplied times the value from the (I-1)th array before adding it in the Ith array. For example, the node (N=0, I=1) is equal to $X_0 + W^0 X_4$. In the program, each successive array is computed before another is begun.

In program listing the symbol A indicates the storage address of the length of the Node Run.

The curly brace "A" indicates a Node Run of lnegth $\frac{NUM}{2^I}$ = 4 in the first array.

overflow would otherwise occur. The test for overflow is carried out by method number 3), described by Welch (1969).

The operation of "scaling" consists of dividing all members of the array by 2. The number of times scaling has been carried out is saved in location labelled SCNT. At the conclusion of a forward transform calculation the correct value of transform can be found from the array output according to

$$2^{SCNT-M} * (array) \qquad (6)$$

while the inverse transformed values are

$$2^{SCNT} * (array). \qquad (7)$$

5) The program is in the form of a subroutine whose starting address is stored in location 103. Input data must be in the form of (two's complement) signed integers. Output will be in the same form.

a) Calling sequences:

JSR @ 103	; 103 contains 5062 in this listing
<FLAG>	; FLAG=0 for forward transform, 1 for inverse.
<NUM>	; Number of complex data input
<M>	; $M=\log_2$ (NUM)
<ALPHA>	; Lowest address of bugger for storage of input and output arrays (see b).
<SCNT>	; Scaling constant (see below)
<RETURN>	;

As presently assembled the subroutine can operate on an array of 1024 (2^{10}) complex values. The subroutine itself requires 1177 octal locations of which six are in page zero. A block of 400 (octal) at the

beginning of the subroutine contains a table of cosines. The cosines are stored in a form such that an integer 40,000 (octal) in storage would represent a cosine value of unity. This table must be changed in order to accommodate an array of input data longer than 1024. Two other routines would need change also, FFTRA, FFTCC. These rearrange the order of cosines if a shorter than maximum length array is to be transformed.

b) Upon entry to the subroutine, the input data array is contained in a buffer beginning at the arbitrary address ALPHA. After return, the transformed results will be found to have replaced the input array. Let $X_n + i\,Y_n$ be the input data expressed as real and imaginary parts, and $U_k + i\,V_k$ be the complex results. (If this is a forward transform then

$$X_n + i\,Y_n = f(NT)$$

$$F(K\Omega) = \frac{1}{(NUM)} * (2^{SCNT}) * (U_k + i\,V_k)$$

according to equations 1 and 6).

The arrays are stored in the following order:

INPUT: $X_0, X_1, \ldots, X_{NUM-1}, Y_0, Y_1, \ldots, Y_{NUM-1}$.

OUTPUT: $U_0, U_1, \ldots, U_{NUM-1}, V_0, V_1, \ldots, V_{NUM-1}$.

6) This program takes about 15 seconds to calculate the DFT of an array of 1024 complex input data. Time was measured on a NOVA without hardware multiply option.

7) Suppose we have a series $_x$ of length 2N, twice that permitted in our program. We can follow the method shown in equations 3-5 to arrive at a DFT of the double length series. Thus (modifying the notation of

3-5) if g_ℓ and h_ℓ represent the even and odd points of x_ℓ then the DFT of x is eq. 5 or

$$X_k = \begin{cases} G_k + H_k \exp\left(-\frac{i\pi k}{N}\right) & , \text{ if } 0 \leqslant k < N \\ G_k + H_{k-N} \exp\left(-\frac{i\pi k}{N}\right) & , \text{ if } N \leqslant k < 2N. \end{cases}$$

Indeed, if the double length series is purely real then we can compute G and H simultaneously by constructing a normal length complex series f_ℓ in which g_ℓ and h_ℓ are the real and imaginary parts respectively. The DFT F_k of the complex series can be decomposed into the required quantities G_n and H_n because

$$\text{Rl}(F_k) = \sum_{o}^{N-1} \left[g_\ell \cos\frac{(2\pi \ell k)}{N} + h_\ell \sin\frac{(2\pi \ell k)}{N} \right],$$

$$\text{Im}(F_k) = \sum_{o}^{N-1} \left[-g_\ell \sin\frac{(2\pi \ell k)}{N} + h_\ell \cos\frac{(2\pi \ell k)}{N} \right].$$

From these relations we find that

$$G_k = 1/2\ \text{Rl}(F_k + F_{n-k}) + \frac{i}{2}\ \text{Im}(F_k - F_{n-k})$$

$$H_k = 1/2\ \text{Im}(F_k + F_{n-k}) - \frac{i}{2}\ \text{Rl}(F_k - F_{n-k}).$$

Often the DFT of a double length series is needed to calculate the spectrum. In this case we need the absolute square of the DFT:

$$|X_n|^2 = |G_k|^2 + |H_k|^2 + \text{Rl}\left[G_k\ H_k^*\ \exp\left(-\frac{\pi k}{N}\right) \right].$$

The right hand term can be represented in terms of real and imaginary parts of F_k as follows:

$$(\text{Rl}\ F_k\ \text{Im}\ F_{n-k} + \text{Rl}\ F_{n-k}\ \text{Im}\ F_k) \cos\left(\frac{\pi k}{N}\right)$$

$$+ 1/2 \left[(\text{Rl}\ F_k)^2 - (\text{Im}\ F_k)^2 - (\text{Rl}\ F_{n-k})^2 + (\text{Im}\ F_{n-k})^2 \right] \sin\left(\frac{\pi k}{N}\right).$$

FFT PROGRAM ANNOTATION

PART I: THE INITIALIZING SUBROUTINES

```
;"FFTST"-This subroutine gets the forward-or-reverse-transform
;flag, the number of data points in the input array, NUM, the
;log (base 2) of NUM, M, and the initial location, ALPHA, of
;the data buffer. In addition, the routine clears necessary
;locations so that FFT is reentrant.

FFTST: LDA Ø,Ø,3          ;get forward(=Ø)-or-reverse(=1)-trans-
       STA Ø,FLAG         ;form-flag and store it.(See call. seq.).
       LDA Ø,1,3          ;get number of data points and store
       STA Ø,NUM          ;it.
       MOVZR Ø,Ø          ;save NUM/4 for use in "FFTCC"
       MOVZR Ø,Ø          ;NUM/4=# of cosines needed for specific
       STA Ø,NUM4         ;value of NUM.
       STA Ø,BACK         ;
       LDA Ø,2,3          ;get log (base 2) of NUM and
       STA Ø,M            ;store it.
       LDA Ø,3,3          ;get the initial location of the input
       STA Ø,ALPHA        ;data array and store it.
       LDA Ø,FOUR         ;=4.
       ADD Ø,3            ;get the location for storing the value of
       STA 3,GMBP         ;the scaling count.(See write up.).
       SUB 1,1            ;clear AC1.
       STA 1,SCNT         ;
       STA 1,SPF          ;
       JMP FFTCC          ;


;"FFTCC"-This subroutine arranges the cosine table so that
;the cosines needed (as determined by the value of NUM) are
;in sucessive order starting at location BETA.

FFTCC: LDA 1,MM           ;MM=13. It is designed so that when
       LDA Ø,M            ;it is subtracted from M, the result
                          ;= M minus the number of times a binary
                          ;units digit must be shifted left to
                          ;arrive at the distance between sucess-
                          ;sive desired cosines in the original
                          ;array.
       SUBOR# Ø,1,SNR     ;Is any change necessary?
       JMP @GFFIN         ;No, continue.
       LDA 3,BETA         ;Yes, compute the distanced or
       STA 3,2Ø           ;"SPACING FACTOR".
       STA 3,21           ;
       SUB 1,Ø            ;M-MM=ACØ.
       SUBZL 1,1          ;AC1=1.
       INC Ø,Ø,SZR        ;
       MOVZL 1,1,SKP      ;
       MOV Ø,Ø,SKP        ;
       JMP .-3            ;
       STA 1,SPF          ;save this "SPACING FACTOR".
       ADD 1,3            ;Rearrange the cosine array in order
       LDA Ø,Ø,3          ;to accomodate the input series length.
```

```
        LDA 2,-2Ø                ;
        STA Ø,@21                ;
        STA 2,Ø,3                ;
        DSZ BACK                 ;
        JMP .-6                  ;
        JMP @FFTIN               ;Go to FFTNC.
```

PART II: THE NUMERICAL CALCULATIONS SUBROUTINES

```
;"CALC"-This subroutine takes the values for N and A as
;provided by FFTNC (see annotation for FFTNC) and retrieves
;the proper values for COS and SIN (via FFTAG), and the
;values for the complex numbers in the array as determined
;by N and A. THE routine uses FFTNT and MULt to multiply
;the complex number X sub (N+A) times (COS ∓ iSIN). In
;addition, it calls FFTAT to test for overflow and scale if
;necessary.

CALC:   STA 3,RETN               ;save return.
        LDA 1,N                  ;"NODE INDEX".
        JSR FFTAG                ;determine proper value for SIN & COS.
        LDA 2,ALPHA              ;
        LDA 1,N                  ;
        ADD 1,2                  ;AC2=address of Re(X sub N).
        LDA Ø,Ø,2                ;
        STA Ø,TEMP1              ;temp. storage of Re(X sub N).
        LDA 3,NUM                ;
        LDA 1,A                  ;"NODE RUN" (See "FFTNC").
        STA 2,SAV1               ;
        ADD 2,3                  ;AC3=address of Im(X sub N).
        [illegible]              ;
        STA 3,SAV2               ;
        ADD 1,2                  ;AC2=address of Re(X sub (N+A)).
        STA 2,SAV3               ;
        ADD 1,3,                 ;AC3=address of Im(X sub (N+A)).
        STA 3,SAV4               ;
        LDA 1,COS                ;signed value of the cosine
        JSR FFTNT                ;test to see if neg  If so, inc ASF.
        MOV 1,2                  ;Move abs. COS to AC2.
        LDA 1,@SAV3              ;get Re(X sub (N+A)).
        JSR FFINT                ;
        JST MULT                 ;ACØ=Re(X sub N+A) times COS.
        STA Ø,@SAV1              ;save temporarily.
        LDA 1,SIN                ;signed value of the sine.(FLAG=Ø +sine).
        JSR FFTNT
        MOV 1,2
        LDA 1,@SAV4              ;get Im(X sub (N+A)).
        JSR FFINT
        JSR MULT                 ;ACØ=Im(X sub N+A)). times SIN.
        LDA 2,@SAV1
        JSR FFTAT                ;get into AC3/AC2 Re(X sub N) +/-
        TEMP1                    ;Re(X sub(N+A)) times COS + Im(X sub (N+A)
                                 ;times SIN.
        STA 3,@SAV1              ;=new value for Re(X sub N).
```

```
        STA 2,TEMP3             ;temp. storage for Re(X sub (N+A)).
        LDA 1,COS               ;
        JSR FFTNI               ;
        MOV 1,2                 ;
        LDA 1,@SAV4             ;
        JSR FFINT               ;
        JSR MULT                ;AC0=In(X sub (N+A)) times COS.
        STA 0,@SAV4             ;temp. storage.
        LDA 1,SIN               ;
        NEG 1,1                 ;
        JSR FFTNT               ;
        MOV 1,2                 ;
        LDA 1,@SAV3             ;
        JSR FFIN.I              ;
        JSR MULT                ;AC0=Re(X sub (N+A) times -SIN.
        LDA 2,@SAV4             ;
CCONT:  LDA 3,@SAV2             ;put In(X sub N) into TEMP1.
        STA 3,TEMP1             ;
        JSR FFTAT               ;get into AC3/AC2 In(X sub N) +/-
        TEMP1                   ;In(X sub (N+A)) times COS + Re(X sub (N+A))
                                ;times -SIN.
        STA 3,@SAV2             ;store new value into array.
        STA 2,@SAV4             ;   "    "    "    "    "
        LDA 0,TEMP3             ;get new value that was stored tempor-
        STA 0,@SAV3             ;arily into the array.
        JMP @RETN               ;


        ;"FFTAT"-This subroutine adds values sent to it by "CALC",
        ;tests to see if overflow has occurred, and "scales" the
        ;data array if overflow has occurred. (note the start of this
        ;routine is towards the end of the listing.)

AZERO:  MOV 0,2                 ;Since AC2=0, the final values for
        MOV 0,3                 ;AC2 & AC3 = AC0.
        JMP @ATRET              ;

CPDD:   ADD 0,2                 ;opposite signs  no overflow. Add & cont.
        MOVZL 1,1,SZC           ;Is this 1st sum?
        JMP CNT2                ;No, go calculate final values.
TCONT:  LDA 0,@-1,3             ;get value from "CALC" (Re or In(N)).
        SUBZR 1,1               ;set bit-15 to indicate that the 1st
                                ;sum is completed.
        MOVZL# 0,0,SZC          ;Is new op. neg?
        INC 1,1                 ;yes, set bit-0=1
        NEG 2,3,SNR             ;remember! we want (X sub N) - (X sub (N+A))
                                ;times (COS -iSIN).
        JMP AZERO
        MOVZL# 2,2,SZC          ;

        INC 1,1
        MOVZR# 1,1,SNC          ;Are Ac2 & AC0 of the same sign?
        JMP TSAME               ;Yes, add and test for overflow.
        ADD 0,2                 ;No, sum since overflow impossible.
        MOV 2,1                 ;rearrange AC's so that we can sum & test.
        MOV 3,2
        MOV 1,3
        SUBZR 1,1               ;
        MOVCR 1,1               ;set bits 15 & 14.
        JMP TSAME               ;
```

```
T1:     MOVR 2,2,SZC        ;shift AC2 right and round off.
        INC 2,2
        STA 2,.STOR1        ;save value temp.
        MOVL 1,1
        MOVL 1,1,SNC        ;Is this the 2nd or 3rd sum?
        JMP FT2             ;It's the 2nd.
        MOV 3,Ø             ;set up the contents of AC3 to be rounded.
        JSR BSCL            ;ROUND off the contents of ACØ & scale the
                            ;data array.
        LDA 3,STOR2
        LDA 2,STOR1
        JMP @ATRET

FT2:    ADD 3,Ø             ;get 3rd sum in ACØ.
        JSR BSCL
        LDA 2,STOR2
        LDA 3,STOR1
        JMP @ATRET

HT2:    MOVZL 1,1,SZC       ;Is the 3rd sum completed.
        JMP @ATRET          ;Yes.
        ADD Ø,3             ;No, get it.
        MOV 3,1             ;rearrange AC's for proper interpret-
        MOV 2,3             ;ation by "CALC".
        MOV 1,2
        JMP @ATRET

BSCL:   STA 3,BACK          ;save return.
        JSR ROUND           ;round off ACØ.
        STA Ø,STOR2         ;
        LDA 2,ALPHA         ;get beginning location of data array.
        LDA 1,NUM           ;
        MOVZL 1,1           ;2 times NUM = number of points to be
        NEG 1,1             ;scaled.
SCLOP:  LDA Ø,Ø,2           ;get data into ACØ.
        JSR ROUND           ;divide by 2 & round off.
        STA Ø,Ø,2           ;replace it.
        INC 2,2
        INC 1,1,SZR         ;repeat until both the real & imaginary
        JMP SCLOP           ;buffers are scaled.
        ISZ SCNT            ;increment "SCALE COUNT".
        JMP @BACK

ROUND:  MOVZL# Ø,Ø,SZC      ;Is the number pos. or neg?
        MOVOR Ø,Ø,SKP       ;shift right placing a 1 in bit-15.(neg.)
        MOVZR Ø,Ø           ;pos.
        MOV Ø,Ø,SZC         ;interogate carry in order to round off.
        INC Ø,Ø
        JMP Ø,3

FFTAT:  INC 3,3             ;start routine here. Set up t  return
        STA 3,ATRET         ;at JSR+2.

TSLOP:  MOV Ø,Ø,SNR         ;handle special case for op.=Ø.
        JMP OPDD
        MOVZL# 2,2,SZC      ;Is AC2 neg?
        INC 1,1             ;Yes.
        MOVZR# 1,1,SZC      ;Are ops. same sign or opposite?
        JMP OPDD            ;opposite.
```

```
ISAME:  ADCZ Ø,2,SZC            ;add & start test. If op. neg. don't skip.
        INC 1,1
        MOVZL# 2,2,SZC          ;Is bit-15=1?
        INC 1,1                 ;Yes.
        MOVZR# 1,1,SNC          ;Overflow if ops. are pos. & bit-15=1
                                ;or if ops. are neg. & bit-15=Ø.
        JMP OPDL+1

FAIL:   MOVZL# 1,1,SZC          ;Is this 1st sum?
        JMP FT1                 ;No.
        MOVR 2,2,SZC            ;Yes, round off partial sum.
        Inc 2,2
        STA 2,STOR1
        LDA Ø,4-1,3             ;get Re or Im(X sub N).
        JSR BSCL                ;SCALE data array.
        LDA 2,STOR1             ;reset AC's for summing. Overflow could
        LDA Ø,STOR2             ;occur again.
        JSR TCONT+1             ;cont.
```

. . .

```
;"FFTAG"-This is a subroutine which chooses the proper value
;for SIN and COS given N.The cosines are stored at the beginning
;of the program in sequential order. SIN is arrived at by
;using trig. identities.(See also "FFTCC".)

FFTAG:  STA 3,MSAV              ;save return.
        LDA Ø,I                 ;get the value for I (See "FFTNC").
        LDA 2,M
        SUB 2,Ø,SNR             ;=-(M-I).
        JMP MID                 ;last array. No shifting necessary.
        MOVZR 1,1               ;shift N right (M-I) times.
        INC Ø,Ø,SZR
        JMP .-2
MID:    JSR FFTRV               ;reverse the last M digits of N (shifted).
        LDA Ø,NUM4              ;NUM4=NUM/4.
        MOV Ø,2
        SUBZ 1,Ø,SZC            ;Is N (reversed & shifted) in 1st quadrant?
        JMP .+7                 ;Yes.
        ADD 2,0                 ;no, find its value via its supplement.
        LDA 2,@GETA             ;(get initial address of cosine table).
        ADD Ø,2                 ;find its supplement.
        LDA Ø,Ø,2
        NEG Ø,Ø                 ;and negate it.
        JMP .+4
        LDA 2,@GETA             ;yes, get its value and store it in
        ADD 1,2                 ;location COS
        LDA Ø,Ø,2               ;
        STA Ø,COS               ;store it in location COS.
        LDA Ø,NUM4              ;sine (X)=cosine (X-π/2).
        ADCZ# 1,Ø,SZC
        SUB 1,Ø,SKP
        SUB Ø,1,SKP
        MOV Ø,1
        LDA 2,@GETA
        ADD 1,2
        LDA Ø,Ø,2
```

. . .

```
        LDA 2,FLAG              ;FLAG is forward-or-reverse transform
        MOVZR# 2,2,SZC          ;flag. Ø-forward & 1-reverse
        NEG Ø,Ø
        STA Ø,SIN
        JMP @MSAV               ;return


        ;"FFTRV"-this is a subroutine which reverses the last M
        ;binary digits of a word in AC1.

FFTRV:  LDA 2,M                 ;get M
        MOV 1,Ø
        SUB 1,1                 ;clear AC1.
        NEG 2,2                 ;=-M.
        MOVZR Ø,Ø               ;move last digit of number into carry.
        MOVL 1,1                ;move carry to last digit of AC1.
        INC 2,2,SZR             ;do this M times.
        JMP .-3
        JMD Ø,3                 ;return


        ;"FFINT"-This is a subroutine that sets AC1 = to its absolute
        ;value and keeps track of its sign.

FFINT:  MOVZL# 1,1,SNC          ;positive?
        JMP Ø,3                 ;yes, return
        NEG 1,1                 ;No, negate.
        ISZ ASF                 ;inc. sign flag.
        JMP Ø,3


        ;"MULT"-This is a subroutine that multiplies two unsigned
        ;numbers and then gives them a sign as indicated by the sign
        ;flag, ASF.

MULT:   SUBC Ø,Ø                ;clear ACØ.
        STA 3,MSAV              ;save return.
        LDA 3,M2Ø               ;get count (tells how many bits in the
                                ;multiplicand=2Ø(octal)).
MLOOP:  MOVR 1,1,SNC            ;see the NOVA manual for a detailed
        MOVR Ø,Ø,SKP            ;explanation of how the MULT routine
        ADDZR 2,Ø               ;works.
        INC 3,3,SZR
        JMP MLOOP
        MOVCL Ø,Ø               ;the multiplied numbers are shifted left
        MOVZL 1,1               ;twice to fit the special format of the
        MOVL Ø,Ø                ;sine and cosine terms, i. e. 4Ø000=1 (oct
                                ;for sine & cosine.
        ;this part assigns a sign to the product.

        LDA 1,ASF               ;check the sign flag to see if inc.?
        MOVZR# 1,1,SNC          ;an even number of times?
        JMP @MSAV               ;Not,negate
        STA 3,ASF               ;clear ASF.
        JMP @MSAV               ;return.
```

PART III: THE INDEX CONTROL SUBROUTINE

```
        ;"FFTNC"-This is a subroutine to control the indices that
        ;control the Fourier Transform calculations.

FFTNC:  LDA 1,N                 ;Set the number of arrays to be
        STA 1,ACNT              ;calculated(=ACNT) = N.
        SUBZL Ø,Ø               ;Initialize I (so that I=1).
        STA Ø,I                 ;I is the number of the array being
                                ;calculated.
NODS:   SUB 1,1                 ;Initialize N (so that N=Ø).
        STA 1,N                 ;N is the "NODE INDEX".

        :                       ;Determine the length of a "NODE RUN"=NUM/2 exp I.

AGET:   LDA Ø,1
        LDA 2,NUM
        NEG Ø,Ø
        MOVZR 2,2               ;divide NUM by 2 exp I.
        INC Ø,Ø,SZR
        JMP .-2
        STA 2,A                 ;save as the contents of loc. A.
        STA 2,NCNT              ;Set "NODE COUNT" for calc. purposes.

ROAD:   JSR @GCLAC              ;calc. a pair of nodes (elts. of the Ith
                                ;array), X sub N & X sub N+A.
        ISZ N                   ;
        DSZ NCNT                ;decrement node count & test if finished
                                ;an array.
        JMP ROAD                ;Not finished, calc. next pair, i. e.
                                ;X sub (N+1) & X sub (N+A+1).
        LDA 2,A                 ;finished, skip all the A sub (N+A) terms
        STA 2,NCNT              ;that have already been calculated &
        LDA 1,N                 ;test whether the entire array has
        ADD 2,1                 ;been completed.
        STA 1,N                 ;=(N-1) + A +1.
        SUB 1,2,SZR
        JMP ROAD                ;No, not finished with the array. Calc.
                                ;next "NODE RUN".
        ISZ I                   ;finished with the Ith array. Set I=(I+1).
        DSZ ACNT                ;Finished the transform?
        JMP NODS                ;No, calc. the (I+1)th array
        JMP OUT                 ;Yes, rearrange the transform array & the
                                ;the cosine array.
```

PART IV: THE UNSCRAMBLING SUBROUTINES

```
;"OUT"-This is a subroutine which unscrambles the I=N array so as to
        ;have the Fourier Transform coefficients in their proper order.

OUT:    LDA 1,PHASE             ;"PHASE" is a cycling device.
        STA 1,2Ø
        LDA 1,ALPHA
        STA 1,ALPHB
        MOV @,V,SKP
```

```
        JMP @GFFRA              ;Go to "GFFRA" to restore the cosine array.
        SUB 1,1                 ;clear N
        STA 1,N
        LDA 1,NUM
        STA 1,NCNT
CYCLE:  LDA 1,N                 ;Get N (ranges Ø,. .. ...,NUM-1).
        JSR FFTHV               ;reverse the last M digits of N.
        LDA 2,ALPHB             ;get array starting location
        LDA 3,N
        SUBZ# 1,3,SZC           ;Is M N(reversed).
        JMP EQUAL               ;ns, the data is in its correct place.
        ADD 2,3                 ;yes, get proper location of A(N) (data in
                                ;array,A, related to N).
        ADD 1,2                 ;get location of N(reversed).
        LDA 1,Ø,2               ;switch their contents.
        LDA Ø,Ø,3
        STA Ø,Ø,2
        STA 1,Ø,3

EQUAL?  ISZ N
        DSZ NCNT                ;Through array?
        JMP CYCLE               ;No, continue to unscramble.
        LDA 2,ALPHB             ;Yes, get initial location of imag. parts
                                ;array.
        LDA 1,NUM
        ADD 1,2
        STA 2,ALPHB
        JMP @2Ø

PHASE:  OUT+3.
```

;"FFTRA"-This is a subroutine to restore the cosine array to
;its original order.

```
FFTRA:  LDA Ø,SCNT              ;get the number of times that "scaling"
        STA Ø,@GMBP             ;has occurred in this transform & store
                                ;the value in the calling seq. at return-1.
        ISZ GMBP                ;Set loc. to return to.
        LDA Ø,SPF               ;get "SPACING FACTOR".
        MOVZR# Ø,Ø,SNR          ;Test to see if cosine array unchanged.
        JMP @GMBP               ;Done, return.
        LDA 2,BETA              ;get initial loc. of cosine array.
        LDA 3,MAX4              ;get final location of cosine array.
        LDA 1,NUM4              ;get number of cosines in the ordered,
                                ;altered array.
        ADD 1,2                 ;=final loc.-1 of ordered, altered array.
        INC 2,2
        LDA 1,SPF
        STA 2,3Ø                ;set up autodecrementing locations.
        STA 2,31
        LDA Ø,Ø,3               ;get last unordered cosine.
        LDA 2,@3Ø               ;get last ordered cosine.
        STA Ø,@31               ;Switch their locations.
        STA 2,Ø,3
        SUB 1,3                 ;come down from the top to the next
                                ;displaced cosine.
        DSZ NUM4                ;Continue this process NUM/4 times.
```

```
JMP @GRBP          ;Finished with the Fourier Transform.
                   ;return to the main program.
```

PART V: THE SYMBOLS AND THEIR USES

```
A          ;"NODE RUN" length.
           ;=NUM/(2 exp I).
ACNT       ;"ARRAY COUNT", initially=M
AGET       ;"FFINC subroutine.
ALPHA      ;initial loc. of data array
ALPHB      ;cycling device for "OUT"
ASF        ;sign flag.
ATRET      ;return loc. ("FFTAT").
AZERO      ;"FFTAT" subroutine.
BACK       ;temp. loc.
BSCL       ;"FFTAT" subroutine to
           ;scale the data array.
BETA:      ;initial loc. of cos. arr.
CALC       ;central computing sub.
CHI2       ;"FFTAT" sub.
COS        ;cosine storage loc.
CYCLE      ;"OUT" sub.
EQUAL      ;   "     "
FFTAG      ;sub. to find values of
           ;trig. functions given N.
FFTAT:     ;sum & overflow test sub.
FFTCC      ;cosine array ordering sub.
FFTNC      ;index control sub.
FFTNT      ;sign test sub.
FFTRA      ;restores orig. nature of
           ;cosine array.
FFTRV      ;reverses last M digits AC1.
FFTST      ;initializing sub.
FLAG       ;forward-or-reverse trans.
           ;flag. Ø-forward, 1-reverse
FOUR       ;=4.
FT1        ;"FFTAT" sub.
FT2        ;   "     "
GCALC      ;device to get form
           ;"FFTNC" to "CALC".
GFFRA      ;device to get from
           ;"OUT" to "FFTRA".
GFFTN      ;device to get from
           ;"FFTCC" to "FFTNC".
I          ;array index.
M          ;=log (base 2) NUM.
M2Ø        ;used in "MULT"=-2Ø.
MAX4       ;last loc. in cos. array.
MID        ;FFTAG sub.
MLOOP      ;processing loop of "MULT".
MM         ;=13. Used in "FFTCC".
MSAV       ;temp. loc.
MULT       ;multipling sub.
N          ;node index.
NCNT       ;node count.
NODS       ;"FFINC" sub.
NUM        ;data array length.
NUM4       ;=NUM/4.
OFDD       ;"FFTAT" sub.
OUT        ;data array unscramb...
           ;sub.
PHASE      ;cycling device for
           ;"OUT".
RETN       ;return temp. loc.
RGAD       ;"FFINC" sub.
ROUND      ;"FFTAT" sub. to round
           ;off a number in ACØ.
SAV1       ;ALPHA+N.
SAV2       ;ALPHA+N+NUM.
SAV3       ;ALPHA+N+A.
SAV4       ;ALPHA+N+NUM+A.
SCLOP      ;"BSCL" sub.
SCNT       ;scale count (=number
           ;of times the data ar...
           ;was scaled during co...
           ;of transform.
SIN        ;temp. storage of sin.
STOR1      ;temp. storage for ...
STOR2      ;   "      "     "
TCONI      ;"FFTAT" sub.
TEMP1      ;"CALC" temp. loc.
TEMP2      ;   "     "    "
TSAME      ;"FFTAT" sub.
TSLOP      ;   "     "
```

```
        000100  .LOC 100
00100 000000 NUM: 0
00101 000000 ALPHA: 0
00102 000000 M: 0
00103 005062 GFTST: FFTST
00104 000000 FLAG: 0
00105 000000 NUM4: 0

        005001  .LOC 5001
05001 020567 FFTRA: LDA 0,SCNT
05002 042503 STA 0,@GMBP
05003 010502 ISZ GMBP
05004 020424 LDA 0,SPF
05005 101235 MOVZR# 0,0,SNR
05006 002477 JMP @GMBP
05007 030562 LDA 2,BETA
05010 034417 LDA 3,MAX4
05011 024105 LDA 1,NUM4
05012 133000 ADD 1,2
05013 151400 INC 2,2
05014 024414 LDA 1,SPF
05015 050030 STA 2,30
05016 050031 STA 2,31
05017 021400 LDA 0,0,3
05020 032030 LDA 2,@30
05021 042031 STA 0,@31
05022 051400 STA 2,0,3
05023 136400 SUB 1,3
05024 014105 DSZ NUM4
05025 000772 JMP .-6
05026 002457 JMP @GMBP
05027 005000 MAX4: 5000
05030 000000 SPF: 0
05031 005453 GFFTN: FFTNC


05032 024426 FFTCC: LDA 1,MM
05033 020102 LDA 0,M
05034 106655 SUBOR# 0,1,SNR
05035 002774 JMP @GFFTN
05036 034533 LDA 3,BETA
05037 054020 STA 3,20
05040 054021 STA 3,21
05041 122400 SUB 1,0
05042 126520 SUBZL 1,1
05043 101404 INC 0,0,SZR
05044 125121 MOVZL 1,1,SKP
05045 101001 MOV 0,0,SKP
05046 000775 JMP .-3
05047 044761 STA 1,SPF
05050 137000 ADD 1,3
05051 021400 LDA 0,0,3
05052 032020 LDA 2,@20
05053 042021 STA 0,@21
05054 051400 STA 2,0,3
05055 014431 DSZ BACK
05056 000772 JMP .-6
05057 002752 JMP @GFFTN
```

```
05060 000013 MM: 13
05061 000004 FOUR: 4
```

```
;;;
05062 021400 FFTST: LDA 0,0,3
05063 040104 STA 0,FLAG
05064 021401 LDA 0,1,3
05065 040100 STA 0,NUM
05066 101220 MOVER 0,0
05067 101220 MOVER 0,0
05070 040105 STA 0,NUM4
05071 040415 STA 0,BACK
05072 021402 LDA 0,2,3
05073 040102 STA 0,M
05074 021403 LDA 0,3,3
05075 040101 STA 0,ALPHA
05076 020763 LDA 0,FOUR
05077 117000 ADD 0,3
05100 054405 STA 3,GMBP
05101 126400 SUB 1,1
05102 044466 STA 1,SCNT
05103 044725 STA 1,SPF
05104 000726 JMP FFTCC
05105 000000 GMBP: 0
```

```
05106 000000 BACK: 0

05107 111000 AZERO: MOV 0,2
05110 115000 MOV 0,3
05111 002425 JMP @ATRET

05112 113000 OPDD: ADD 0,2
05113 125122 MOVZL 1,1,SZC
05114 000445 JMP CHT2
05115 023777 TCONT: LDA 0,0-1,3
05116 126620 SUBZR 1,1
05117 101132 MOVZL# 0,0,SZC
05120 125400 INC 1,1
05121 154405 NEG 2,3,SNR
05122 000765 JMP AZERO
05123 151132 MOVZL# 2,2,SZC
05124 125400 INC 1,1
05125 125233 MOVZR# 1,1,SNC
05126 000501 JMP TSAME
05127 113000 ADD 0,2
05130 145000 MOV 2,1
05131 171000 MOV 3,2
05132 135000 MOV 1,3
05133 126620 SUBZR 1,1
05134 125240 MOVOR 1,1
05135 000472 JMP TSAME
05136 000000 ATRET: 0
05137 000000 STOR1: 0
05140 000000 STOR2: 0

05141 151202 FT1: MOVR 2,2,SZC
05142 151400 INC 2,2
05143 050774 STA 2,STOR1
05144 125100 MOVL 1,1
05145 125123 MOVZL 1,1,SNC
05146 000406 JMP FT2
05147 161000 MOV 3,0
05150 004422 JSR BSCL
05151 034767 LDA 3,STOR2
05152 030765 LDA 2,STOR1
05153 002763 JMP @ATRET

05154 163000 FT2: ADD 3,0
05155 004415 JSR BSCL
05156 030762 LDA 2,STOR2
05157 034760 LDA 3,STOR1
05160 002756 JMP @ATRET


05161 125122 CHT2: MOVZL 1,1,SZC
05162 002754 JMP @ATRET
05163 117000 ADD 0,3
05164 165000 MOV 3,1
05165 155000 MOV 2,3
05166 131000 MOV 1,2
05167 002747 JMP @ATRET
05170 000000 SCN1: 0
05171 004400 BETA: 4400
05172 054714 BSCL: STA 3,BACK
```

```
05173 004416 JSR ROUND
05174 040744 STA 0,STOR2
05175 030101 LDA 2,ALPHA
05176 024100 LDA 1,NUM
05177 125120 MOVEL 1,1
05200 124400 NEG 1,1
05201 021000 SCLOP: LDA 0,0,2
05202 004407 JSR ROUND
05203 041000 STA 0,0,2
05204 151400 INC 2,2
05205 125404 INC 1,1,SER
05206 000773 JMP SCLOP
05207 010761 ISE SCNT
05210 002676 JMP @BACK

05211 101132 ROUND: MOVEL# 0,0,SEC
05212 101241 MOVOR 0,0,SKP
05213 101220 MOVER 0,0
05214 101002 MOV 0,0,SEC
05215 101400 INC 0,0
05216 001400 JMP 0,3

05217 175400 FFTAT: INC 3,3
05220 054716 STA 3,ATRET
.....
05221 101005 TSLOP: MOV 0,0,SNR
05222 000670 JMP OPDD
05223 151132 MOVEL# 2,2,SEC
05224 125400 INC 1,1
05225 125232 MOVER# 1,1,SEC
05226 000664 JMP OPDD
05227 113022 TSAME: ADDE 0,2,SEC
05230 125400 INC 1,1
05231 151132 MOVEL# 2,2,SEC
05232 125400 INC 1,1
05233 125233 MOVER# 1,1,SNC
05234 000657 JMP OPDD+1

05235 125132 FAIL: MOVEL# 1,1,SZC
05236 000703 JMP FT1
05237 151202 MOVR 2,2,SEC
05240 151400 INC 2,2
05241 050676 STA 2,STOR1
05242 023777 LDA 0,2-1,3
05243 004727 JSR BSCL
05244 030673 LDA 2,STOR1
05245 020673 LDA 0,STOR2
05246 000650 JMP TCONT+1
```

```
05247 054531 CALC: STA 3,RETN
05250 024547 LDA 1,N
05251 004502 JSR FFTAG
05252 030101 LDA 2,ALPHA
05253 024544 LDA 1,N
05254 133000 ADD 1,2
05255 021000 LDA 0,0,2
05256 040473 STA 0,TEMP1
05257 034100 LDA 3,NUM
05260 024540 LDA 1,A
05261 050463 STA 2,SAV1
05262 157000 ADD 2,3
05263 054462 STA 3,SAV2
05264 133000 ADD 1,2
05265 050461 STA 2,SAV3
05266 137000 ADD 1,3
05267 054460 STA 3,SAV4
05270 024452 LDA 1,COS
05271 004531 JSR FFTNT
05272 131000 MOV 1,2
05273 026453 LDA 1,@SAV3
05274 004526 JSR FFTNT
05275 004533 JSR MULT
05276 042446 STA 0,@SAV1
05277 024444 LDA 1,SIN
05300 004522 JSR FFTNT
05301 131000 MOV 1,2
05302 026445 LDA 1,@SAV4
05303 004517 JSR FFTNT
05304 004524 JSR MULT
05305 032437 LDA 2,@SAV1
05306 004711 JSR FFTAT
05307 005351 TEMP1
05310 056434 STA 3,@SAV1
05311 050441 STA 2,TEMP3
05312 024430 LDA 1,COS
05313 004507 JSR FFTNT
05314 131000 MOV 1,2
05315 026432 LDA 1,@SAV4
05316 004504 JSR FFTNT
05317 004511 JSR MULT
05320 042427 STA 0,@SAV4
05321 024422 LDA 1,SIN
05322 124400 NEG 1,1
05323 004477 JSR FFTNT
05324 131000 MOV 1,2
05325 026421 LDA 1,@SAV3
05326 004474 JSR FFTNT
05327 004501 JSR MULT
05330 032417 LDA 2,@SAV4
05331 036414 LDA 3,@SAV2
05332 054417 STA 3,TEMP1
05333 004664 JSR FFTAT
05334 005351 TEMP1
05335 056410 STA 3,@SAV2
05336 052411 STA 2,@SAV4
05337 020413 LDA 0,TEMP3
05340 042406 STA 0,@SAV3
05341 002407 JMP @RETN
```

```
---
???
   05342 000000 COS: 0
   05343 000000 SIN: 0
   05344 000000 SAV1: 0
   05345 000000 SAV2: 0
   05346 000000 SAV3: 0
   05347 000000 SAV4: 0
   05350 000000 RETN: 0
   05351 000000 TEMP1: 0
   05352 000000 TEMP3: 0

   05353 054476 FFTAG: STA 3,MSAV
   05354 020540 LDA 0,I
   05355 030102 LDA 2,M
   05356 142405 SUB 2,0,SNR
   05357 000404 JMP MID
   05360 125220 MOVER 1,1
   05361 101404 INC 0,0,SER
   05362 000776 JMP .-2
   05363 004532 MID: JSR FFTRV



   05364 020105 LDA 0,NUM4
   05365 111000 MOV 0,2
   05366 122422 SUBZ 1,0,SZC
   05367 000407 JMP .+7
   05370 143000 ADD 2,0
   05371 032433 LDA 2,@GETA
   05372 113000 ADD 0,2
   05373 021000 LDA 0,0,2
   05374 100400 NEG 0,0
   05375 000404 JMP .+4
   05376 032423 LDA 2,@GETA
   05377 133000 ADD 1,2
   05400 021000 LDA 0,0,2
   05401 040741 STA 0,COS
   05402 020105 LDA 0,NUM4
   05403 122032 ADCZ# 1,0,SZC
   05404 122401 SUB 1,0,SKP
   05405 106401 SUB 0,1,SKP
   05406 105000 MOV 0,1

   05407 032412 LDA 2,@GETA
   05410 133000 ADD 1,2
   05411 021000 LDA 0,0,2
   05412 030104 LDA 2,FLAG
   05413 151232 MOVER# 2,2,SZC
   05414 100400 NEG 0,0
   05415 040726 STA 0,SIN
   05416 002433 JMP @MSAV
   05417 000000 N: 0
   05420 000000 A: 0
   05421 005171 GETA: BETA
```

```
05422 125133 FFTNT: MOVEL# 1,1,SNC
05423 001400 JMP 0,3
05424 124400 NEG 1,1
05425 010402 ISE ASF
05426 001400 JMP 0,3

05427 000000 ASF: 0
05430 102460 MULT: SUBC 0,0
05431 054420 STA 3,MSAV
05432 034420 LDA 3,M20
05433 125203 MLOOP: MOVR 1,1,SNC
05434 101201 MOVR 0,0,SKP
05435 143220 ADDER 2,0
05436 175404 INC 3,3,SER
05437 000774 JMP MLOOP
05440 101160 MOVCL 0,0
05441 125120 MOVEL 1,1
05442 101100 MOVL 0,0
05443 024764 LDA 1,ASF
05444 125233 MOVER# 1,1,SNC
05445 002404 JMP @MSAV
05446 100400 NEG 0,0
05447 054760 STA 3,ASF
05450 002401 JMP @MSAV
05451 000000 MSAV: 0
05452 177760 M20: -20
```

```
***
05453 024102 FFTNC: LDA 1,M
05454 044435 STA 1,ACNT
05455 102500 SUBEL 0,0
05456 040436 STA 0,I



05457 126400 NODS: SUB 1,1
05460 044737 STA 1,N



05461 020433 AGET: LDA 0,I
05462 030100 LDA 2,NUM
05463 100400 NEG 0,0
05464 151220 MOVER 2,2
05465 101404 INC 0,0,SER
05466 000776 JMP .-2
05467 050731 STA 2,A
05470 050422 STA 2,NCNT



05471 006422 ROAD: JSR @GCALC
05472 010725 ISE N
05473 014417 DSE NCNT
05474 000775 JMP ROAD
05475 030723 LDA 2,A
05476 050414 STA 2,NCNT
05477 024720 LDA 1,N
05500 147000 ADD 2,1
05501 044716 STA 1,N
05502 030100 LDA 2,NUM
05503 132404 SUB 1,2,SER
05504 000765 JMP ROAD
05505 010407 ISE I
05506 014403 DSE ACNT
05507 000750 JMP NODS
05510 000416 JMP OUT

05511 000000 ACNT: 0
05512 000000 NCNT: 0
05513 005247 GCALC: CALC
05514 000000 I: 0
05515 030102 FFTRV: LDA 2,M
05516 121000 MOV 1,0
05517 126400 SUB 1,1
05520 150400 NEG 2,2
05521 101220 MOVER 0,0
05522 125100 MOVL 1,1
05523 151404 INC 2,2,SER
05524 000775 JMP .-3
05525 001400 JMP 0,3
```

```
05526 024437 OUT: LDA 1,PHASE
05527 044020 STA 1,20
05530 024101 LDA 1,ALPHA
05531 044433 STA 1,ALPHB
05532 101001 MOV 0,0,SKP
05533 002433 JMP @GFFRA
05534 126400 SUB 1,1
05535 044662 STA 1,N
05536 024100 LDA 1,NUM
05537 044753 STA 1,NCNT

05540 024657 CYCLE: LDA 1,N
05541 004754 JSR FFTRV

05542 030422 LDA 2,ALPHB

05543 034654 LDA 3,N
05544 136432 SUBE# 1,3,SZC
05545 000407 JMP EQUAL
05546 157000 ADD 2,3
05547 133000 ADD 1,2
05550 025000 LDA 1,0,2
05551 021400 LDA 0,0,3
05552 041000 STA 0,0,2
05553 045400 STA 1,0,3



05554 010643 EQUAL: ISE N
05555 014735 DSZ NCNT
05556 000762 JMP CYCLE
05557 030405 LDA 2,ALPHB
05560 024100 LDA 1,NUM
05561 133000 ADD 1,2
05562 050402 STA 2,ALPHB
05563 002020 JMP @20
05564 000000 ALPHB: 0


05565 005531 PHASE: OUT+3
05566 005001 GFFRA: FFTRA
```

```
???
          004400  .LOC 4400
    04400 040000  040000
    04401 040000  040000
    04402 037777  037777
    04403 037775  037775
    04404 037773  037773
    04405 037770  037770
    04406 037765  037765
    04407 037761  037761
    04410 037754  037754
    04411 037747  037747
    04412 037741  037741
    04413 037733  037733
    04414 037724  037724
    04415 037714  037714
    04416 037704  037704
    04417 037673  037673
    04420 037661  037661
    04421 037647  037647
    04422 037634  037634
    04423 037621  037621
    04424 037605  037605
    04425 037570  037570
    04426 037553  037553
    04427 037535  037535
    04430 037517  037517
    04431 037500  037500
    04432 037460  037460
    04433 037440  037440
    04434 037417  037417
    04435 037375  037375
    04436 037353  037353
    04437 037330  037330
    04440 037305  037305
    04441 037261  037261
    04442 037235  037235
    04443 037210  037210
    04444 037162  037162
    04445 037134  037134
    04446 037105  037105
    04447 037055  037055
    04450 037025  037025
    04451 036774  036774
    04452 036743  036743
    04453 036711  036711
    04454 036657  036657
    04455 036623  036623
    04456 036570  036570
    04457 036533  036533
    04460 036477  036477
    04461 036441  036441
    04462 036403  036403
    04463 036344  036344
    04464 036305  036305
    04465 036245  036245
    04466 036205  036205
    04467 036144  036144
    04470 036102  036102
    04471 036040  036040
```

04472	035775	035775
04473	035732	035732
04474	035666	035666
04475	035622	035622
04476	035555	035555
04477	035507	035507
04500	035441	035441
04501	035372	035372
04502	035323	035323
04503	035253	035253
04504	035202	035202
04505	035131	035131
04506	035060	035060
04507	035006	035006
04510	034733	034733
04511	034660	034660
04512	034604	034604
04513	034530	034530
04514	034453	034453
04515	034375	034375
04516	034317	034317
04517	034241	034241
04520	034161	034161
04521	034102	034102
04522	034022	034022
04523	033741	033741
04524	033660	033660
04525	033576	033576
04526	033513	033513
04527	033430	033430
04530	033345	033345
04531	033261	033261
04532	033175	033175
04533	033110	033110
04534	033022	033022
04535	032734	032734
04536	032645	032645
04537	032556	032556
04540	032467	032467
04541	032377	032377
04542	032306	032306
04543	032215	032215
04544	032123	032123
04545	032031	032031
04546	031737	031737
04547	031643	031643
04550	031550	031550
04551	031454	031454
04552	031357	031357
04553	031262	031262
04554	031164	031164
04555	031066	031066
04556	030770	030770
04557	030671	030671
04560	030571	030571
04561	030471	030471
04562	030371	030371
04563	030270	030270
04564	030166	030166
04565	030064	030064

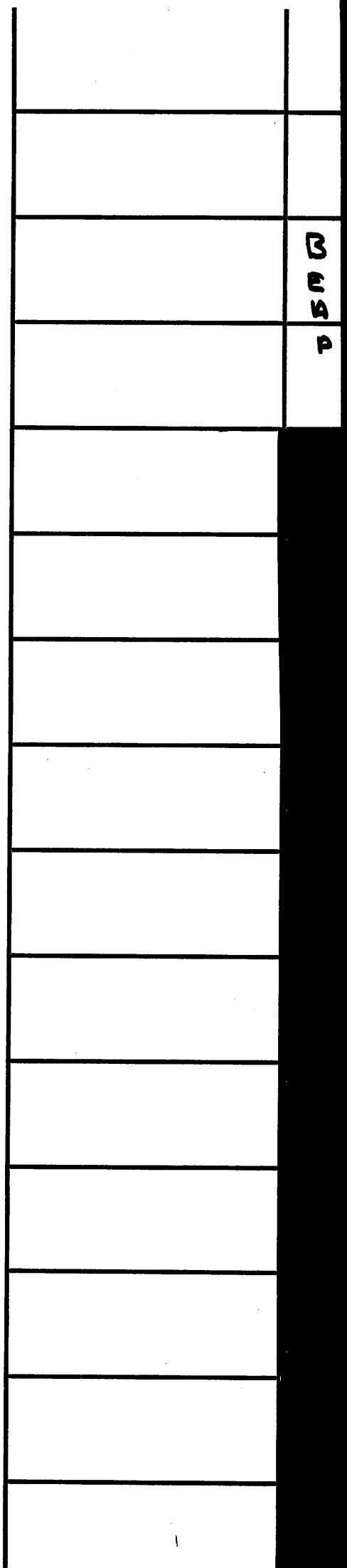
04566	027760	027762
04567	027657	027657
04570	027554	027554
04571	027450	027450
04572	027344	027344
04573	027237	027237
04574	027132	027132
04575	027025	027025
04576	026717	026717
04577	026610	026610
04600	026501	026501
04601	026372	026372
04602	026262	026262
04603	026152	026152
04604	026041	026041
04605	025730	025730
04606	025617	025617
04607	025505	025505
04610	025373	025373
04611	025260	025260
04612	025145	025145
04613	025032	025032
04614	024716	024716
04615	024601	024601
04616	024465	024465
04617	024347	024347
04620	024232	024232
04621	024114	024114
04622	023776	023776
04623	023657	023657
04624	023540	023540
04625	023421	023421
04626	023301	023301
04627	023161	023161
04630	023043	023043
04631	022717	022717
04632	022576	022576
04633	022454	022454
04634	022332	022332
04635	022210	022210
04636	022065	022065
04637	021742	021742
04640	021616	021616
04641	021473	021473
04642	021347	021347
04643	021222	021222
04644	021075	021075
04645	020750	020750
04646	020623	020623
04647	020475	020475
04650	020347	020347
04651	020221	020221
04652	020072	020072
04653	017743	017743
04654	017614	017614
04655	017464	017464
04656	017334	017334
04657	017204	017204
04660	017053	017053
04661	016723	016723

04662	016571	016571
04663	016440	016440
04664	016306	016306
04665	016154	016154
04666	016022	016022
04667	015670	015670
04670	015535	015535
04671	015402	015402
04672	015247	015247
04673	015113	015113
04674	014757	014757
04675	014623	014623
04676	014467	014467
04677	014333	014333
04700	014176	014176
04701	014041	014041
04702	013704	013704
04703	013546	013546
04704	013411	013411
04705	013253	013253
04706	013114	013114
04707	012756	012756
04710	012620	012620
04711	012461	012461
04712	012322	012322
04713	012163	012163
04714	012023	012023
04715	011664	011664
04716	011524	011524
04717	011364	011364
04720	011224	011224
04721	011064	011064
04722	010723	010723
04723	010563	010563
04724	010422	010422
04725	010261	010261
04726	010120	010120
04727	007756	007756
04730	007615	007615
04731	007453	007453
04732	007312	007312
04733	007150	007150
04734	007006	007006
04735	006644	006644
04736	006501	006501
04737	006337	006337
04740	006174	006174
04741	006032	006032
04742	005667	005667
04743	005524	005524
04744	005361	005361
04745	005216	005216
04746	005053	005053
04747	004707	004707
04750	004544	004544
04751	004401	004401
04752	004235	004235
04753	004071	004071
04754	003726	003726
04755	003562	003562

```
04756 003416 003416
04757 003252 003252
04760 003106 003106
04761 002742 002742
04762 002576 002576
04763 002432 002432
04764 002265 002265
04765 002121 002121
04766 001755 001755
04767 001610 001610
04770 001444 001444
04771 001300 001300
04772 001133 001133
04773 000767 000767
04774 000622 000622
04775 000456 000456
04776 000311 000311
04777 000145 000145
05000 000000 000000
                .END
```

A	005420
ACNT	005511
AGET	005461
ALPHA	000101
ALPHB	005564
ARET	005160
ASF	005427
ATRET	005136
AZERO	005107
BACK	005106
BETA	005171
BSCL	005172
CALC	005247
CFTST	000103
CHT2	005161
COS	005342
CYCLE	005540
EQUAL	005554
FAIL	005235
FFTAG	005353
FFTAT	005217
FFTCC	005032
FFTNC	005453
FFTNT	005422
FFTRA	005001
FFTRV	005515
FFTST	005062
FLAG	000104
FOUR	005061
FT1	005141
FT2	005154
GCALC	005513
GETA	005421
GFFRA	005566
GFFTN	005031
GMBP	005105
I	005514
M	000102
M20	005452
MAX4	005027
MID	005363
MLOOP	005433
MM	005060
MSAV	005451
MULT	005430
N	005417
NCNT	005512
NODS	005457
NUM	000100
NUM4	000105
OPDD	005112
OUT	005526
PHASE	005565
RETN	005350
ROAD	005471
ROUND	005211
SAV1	005344
SAV2	005345
SAV3	005346
SAV4	005347

SCLOP	005201
SCNT	005170
SIN	005343
SPF	005030
STOR1	005137
STOR2	005140
TCONT	005115
TEMP1	005351
TEMP3	005352
TSAME	005227
TSLOP	005221


BEAP

DANMARKS INGENIØRAKADEMI
ELEKTROAFDELINGEN
TELETEKNIK
TELEFON (08) 16 05 22
BADEHUSVEJ 1 A
9000 AALBORG


```
;                  ***** B E A P ****

;         EDITOR/ASSEMBLER PROGRAM
;         (MIN 4K + TTY)
;         PROGRAMMER: ROBERT C. BASKIN
;         DATA GENERAL CORP.
;         DATE SUBMITTED: DEC. 30, 1970

;PROGRAM ENABLES EDITING AND ASSEMBLING FROM
;THE CONTENTS OF A MEMORY BUFFER. THUS, I/O
;TIME IS REDUCED TO SOURCE PROGRAM INPUT
;(TAPE OR KEYBOARD) AND OBJECT TAPE OUTPUT.


;START/RESTART = 00377 (BUFFER RETAINED)

;COMMANDS (DON'T FOLLOW WITH CARRIAGE RETURN)

;   Y     "YANK" TEXT FROM THE TTY READER OR KEYBOARD
;         UP TO AND INCLUDING A FORM FEED.
;   P     "PUNCH"(TYPE) THE CONTENTS OF THE BUFFER.
;         PROGRAM HALTS BEFORE AND AFTER OUTPUT TO
;         ALLOW THE PUNCH TO BE TURNED ON OR OFF IF
;         DESIRED.  PRESS CONTINUE.
;   F     PUNCH 10 INCHES OF NULL TAPE. PROGRAM HALTS
;         BEFORE AND AFTER PUNCHING TO ALLOW PUNCH TO
;         BE TURNED ON AND OFF.  PRESS CONTINUE.
;   T     FOLLOWED BY NUMBER(DECIMAL) AND CR, "TYPES"
;         THE DESIRED LINE.
;   D     FOLLOWED BY NUMBER(DECIMAL) AND CR,
;         "DELETES" THE DESIRED LINE. SUCCEEDING LINE
;         NUMBERS ARE DECREMENTED.
;   I     FOLLOWED BY NUMBER(DECIMAL) AND CR,
;         "INSERTS" A LINE BEFORE THE DESIRED LINE.
;         SUCCEEDING LINE NUMBERS ARE INCREMENTED.
;   A     "ASSEMBLES" PROGRAM IN BUFFER FOR ERRORS.
;   L     ASSEMBLES PROGRAM IN BUFFER FOR "LISTING".
;   O     ASSEMBLES PROGRAM IN BUFFER FOR "OBJECT"
;         TAPE.
;   S     PRINTS USER "SYMBOL" TABLE.

;CONTROL MAY BE RETURNED TO BEAP AT ANY TIME BY
;   RESTARTING BEAP(RESET + START AT 377).

;IN THE EVENT OF A COMMAND LINE TYPING ERROR,
;   TYPE A FORM FEED CHARACTER.
```

```
A 00002  .PATN

;BEAP ERROR CODES:

;   D     DISPLACEMENT ERROR
;   F     FORMAT ERROR
;   M     MULTIPLE DEFINED SYMBOL
;   N     NO  .END  STATEMENT
;   O     BUFFER OVERFLOW
;   U     UNDEFINED SYMBOL

;MAXIMUM BUFFER SIZES:

;         255       LINES
;         3712      CHARACTERS
;         114       USER SYMBOLS(LOADERS RETAINED)
;         149       USER SYMBOLS(MAX)


;HINT:

;         WRITE AND DEBUG PROGRAMS IN LOCATIONS
;                   0      THRU  77      AND
;                   7100   THRU  7600
;         THESE LOCATIONS DO NOT EFFECT BEAP,
;         BINARY, OR BOOTSTRAP LOADERS. BEAP
;         BUFFER IS RETAINED.
;         THEN RELOCATE DEBUGGED PROGRAMS BY
;         RE-ASSEMBLY.
```

```
;ASSEMBLER FORMAT:

;                   LABEL: INSTRUCTION ;COMMENT

;LABEL:             UP TO 3 LETTERS FOLLOWED BY
;                   A COLON.
;COMMENT:           A STRING OF CHARACTERS BETWEEN
;                   A SEMI-COLON AND A CR. (NOT
;                   ASSEMBLED).
;INSTRUCTION:       MAY BE ANY OF THE FOLLOWING:
;                   LABEL=NUMBER
;                   NUMBER
;                   -NUMBER
;                   LABEL
;                   .LOC  NUMBER
;                   .LOC  .+NUMBER
;                   .END
;                   .END  NUMBER
;                   .END  LABEL
;                   OPCODE  OPERANDS
;POSSIBLE OPCODES AND OPERANDS:
;       MRI WITHOUT AC
;                   #       #,X       -#,Y     LABEL
;                   LABEL+#           LABEL-#
;                   .+#       .-#
;       MRI WITH AC
;                   AC,(ANY OF ABOVE OPERANDS)
;       I/O WITHOUT AC
;                   #       LABEL
;       I/O WITH AC
;                   AC,#      AC,LABEL
;       ALI
;                   AC,AC             AC,AC,SKIP
;@(INDIRECT) OR #(NO LOAD) SYMBOL MAY OCCUR
;       BEFORE OR AFTER THE OPCODE, OR
;       BEFORE ANY OPERAND.
;OPERANDS MUST BE SEPARATED FROM EACH OTHER
;       BY AT LEAST ONE COMMA.
;OPCODE MUST BE SEPARATED FROM OPERANDS BY
;       AT LEAST ONE SPACE.


        .FOT
```

DIS

ASS
BLE

START AT 2 OR 3

USERS GROUP LIBRARY
PROGRAM #7 DIS-ASSEMBLER
DATE SUBMITTED: FEBRUARY, 1971
SAN ANTONIO COLLEGE

```
      000001            .LOC 1
00001 001537            INTRT
00002 002004            JMP @4
00003 002004            JMP @4
00004 000400            400
      000050            .LOC 50
00050 001372 .PRTXT:    PRTXT
00051 000000 PC:        0 ;PSEUDO PROGRAM COUNTER
00052 001343 .CR:       CR
00053 001501 .OCTAL:    OCTAL
00054 001207 .ARITH:    ARITH
00055 000635 .LONG:     LONG
00056 001303 .PRINT:    PRINT
00057 000000 INSTR:     0  ;THE CURRENT INSTRUCTION
00060 001353 .SHIFT:    SHIFT
00061 001653 OTBUF:     OTBF
00062 001625 INBUF:     INBF
00063 000000 IBUF:      0       ;CHARACTER COUNT
00064 000020            20      ;MAX COUNT
00065 001675            AREA1   ;INPT PTR-VERY NEXT CHAR
00066 001675            AREA1   ;OTPT PTR-VERY NEXT CHAR
00067 001675            AREA1   ;RESET PTR
00070 001715            AREA1+20  ;RESET TEST
00071 000000            0         ;ACTIVE FLAG - TTI
00072 000000 OBUF:      0
00073 000020            20
00074 001715            AREA2
00075 001715            AREA2
00076 001715            AREA2
00077 001735            AREA2+20
00100 000000            0
00101 001432 .ADR2:     ADR2+1
00102 001437 .ADR3:     ADR3+1
00103 001444 .ADR4:     ADR4+1
00104 001447 .ADR5:     ADR5+1
00105 001735 HDR:       HDR1
      000400            .LOC 400
00400 060277 START:     INTDS
00401 062677            IORST
00402 060110            NIOS TTI
00403 102400            SUB 0,0
00404 040063            STA 0,IBUF
00405 040072            STA 0,OBUF
00406 040100            STA 0,OBUF+6
00407 020067            LDA 0,IBUF+4
00410 040065            STA 0,IBUF+2
00411 040066            STA 0,IBUF+3
00412 020076            LDA 0,OBUF+4
00413 040074            STA 0,OBUF+2
00414 040075            STA 0,OBUF+3
00415 102000            ADC 0,0
00416 040071            STA 0,IBUF+6
00417 060177            INTEN
00420 030105            LDA 2,HDR
00421 006050            JSR @.PRTXT
00422 004416 RPT:       JSR READ
00423 101005            MOV 0,0,SNR
00424 000776            JMP .-2
```

```
00425 024503          LDA 1,MSK8
00426 107404          AND 0,1,SZR
00427 000442          JMP DATA ;ITS A DATA BLOCK
00430 126520          SUBZL 1,1
00431 106405          SUB 0,1,SNR
00432 000537          JMP END ;ITS THE END BLOCK
00433 000520          JMP ERROR ;ITS AN ERROR BLOCK
00434 000007 RSTC:    7
00435 000000          0
00436 000000          0
00437 000000          0
00440 054777 READ:    STA 3,.-1
00441 050775          STA 2,READ-2
00442 044773          STA 1,READ-3
00443 020071          LDA 0,IBUF+6  ;GET TTI ACTIVE FLAG
00444 101004          MOV 0,0,SZR
00445 000410          JMP RD.6
00446 020063          LDA 0,IBUF  ;CHAR COUNT
00447 024765          LDA 1,RSTC ;RESTART CONSTANT
00450 106512          SUBL# 0,1,SZC
00451 000404          JMP RD.6  ;DONT START TTI
00452 060110          NIOS TTI
00453 102000          ADC 0,0
00454 040071          STA 0,IBUF+6 ;SET FLAG ON
00455 060277 RD.6:    INTDS
00456 006061          JSR @OTBUF
00457 000063          IBUF
00460 000405          JMP .+5 ;EMPTY!
00461 024754          LDA 1,READ-3
00462 034754          LDA 2,READ-2
00463 040177          INTEN
00464 002753          JMP @READ-1
00465 000177          INTA
00466 000401          JMP .+1 ;GIVE TIME FOR INTERRUPT
00467 000401          JMP .+1
00470 000765          JMP RD.6
00471 103[illegible] DATA:    MOVS 0,1
00472 004746          JSR READ
00473 107300          ADDS 0,1
00474 044451          STA 1,COUNT ;THE WORD COUNT
00475 006052          JSR @.CR
00476 006052          JSR @.CR
00477 004440          JSR RD2
00500 040051          STA 0,PC ;PSEUDO PROGRAM COUNTER
00501 004436          JSR RD2 ;IGNORE CHECK SUM
00502 004435 LOOP:    JSR RD2 ;GO GET INSTRUCTION
00503 040057          STA 0,INSTR ;AND STORE IT
00504 020051          LDA 0,PC
00505 126400          SUB 1,1 ;SUPPRESS LEAD ZEROS
00506 006053          JSR @.OCTAL ;AS CORE LOCN PRINTS
00507 004423          JSR BLKS ;LEAVE TWO SPACES
00510 020057          LDA 0,INSTR
00511 126000          ADC 1,1
00512 006053          JSR @.OCTAL ;PRINT THE INSTR IN OCTAL
00513 004417          JSR BLKS
00514 020057          LDA 0,INSTR
00515 101113          MOVL# 0,0,SNC
00516 000407          JMP XXX2 ;NOT AN ALC TYPE
00517 006054          JSR @.ARITH ;IT IS AN ALC
00520 006052 XXX:     JSR @.CR
```

```
00521 010051            ISZ PC ;BUMP PROGRAM COUNT
00522 010423            ISZ COUNT ;BUMP WORD COUNT
00523 000757            JMP LOOP ;BLOCK NOT FINISHED
00524 000676            JMP RPT ;GO FIND NEXT BLOCK
00525 006055 XXX2:      JSR @.LONG ;GO PROCESS NON ALC TYPE
00526 000772            JMP XXX
00527 000040 C40:       40
00530 000200 MSK8:      200
00531 000000            0
00532 054777 BLKS:      STA 3,.-1
00533 020774            LDA 0,C40
00534 006056            JSR @.PRINT
00535 006056            JSR @.PRINT
00536 002773            JMP @BLKS-1
00537 054772 RD2:       STA 3,BLKS-1 ;SUBRTN TO READ TWO INPUTS
00540 004700            JSR READ     ;FROM TTI AND BUILD INSTR WORD
00541 105300            MOVS 0,1
00542 004676            JSR READ
00543 123300            ADDS 1,0
00544 002765            JMP @BLKS-1
00545 000000 COUNT:     0
00546 000000 TEMP:      0
00547 000550 ADX3:      .+1
00550 051105 .TXT *ER
00551 047522 RO
00552 000122 R*

00553 040773 ERROR:     STA 0,TEMP ;SUBRTN TO PROCESS AN
00554 006052            JSR @.CR   ;ERROR BLOCK
00555 006052            JSR @.CR
00556 030771            LDA 2,ADX3
00557 006050            JSR @.PRTXT
00560 006052            JSR @.CR
00561 020765            LDA 0,TEMP
00562 006056            JSR @.PRINT
00563 004655            JSR READ
00564 101004            MOV 0,0,SZR
00565 000775            JMP .-3
00566 006052            JSR @.CR
00567 006052            JSR @.CR
00570 000632            JMP RPT
00571 006052 END:       JSR @.CR  ;SUBRTN TO PROCESS END BLOCKS
00572 006052            JSR @.CR  ;AND HALT THE PROGRAM
00573 030423            LDA 2,ADR.5
00574 006050            JSR @.PRTXT
00575 004643            JSR READ
00576 004741            JSR RD2
00577 101112            MOVL# 0,0,SZC
00600 000412            JMP END3
00601 040745            STA 0,TEMP
00602 030421            LDA 2,ADR.6
00603 006050            JSR @.PRTXT
00604 020742            LDA 0,TEMP
00605 126400            SUB 1,1
00606 006053            JSR @.OCTAL
00607 004730            JSR RD2
00610 063077            HALT          See correction
00611 000777            JMP .-1
00612 030416 END3:      LDA 2,ADR7
00613 006050            JSR @.PRTXT
```

```
00614 063077          HALT          See correction
00615 000777          JMP .-1
00616 000617 ADR.5:   .+1
00617 047105 .TXT *EN
00620 020104 D
00621 020040
00622 000000 *

00623 000624 ADR.6:   .+1
00624 047507 .TXT *GO
00625 052040  T
00626 020117 O
00627 000040  *

00630 000631 ADR7:    .+1
00631 040510 .TXT *HA
00632 052114 LT
00633 000040  *

00634 001035          IO
00635 054560 LONG:    STA 3,RET ;SUBRTN TO PROCESS NON ALC TYPES
00636 006060          JSR @.SHIFT
00637 060000          60000
00640 000015          15
00641 101005          MOV 0,0,SNR
00642 000422          JMP JUMP ;MEM REF W/O AC TYPE
00643 101226          MOVZR 0,0,SEZ
00644 002770          JMP @LONG-1 ;ITS AN IO TYPE
00645 030543          LDA 2,ADR.2
00646 103000          ADD 0,0
00647 113000          ADD 0,2
00650 006050          JSR @.PRTXT
00651 020545          LDA 0,BLANK
00652 006056          JSR @.PRINT
00653 006060          JSR @.SHIFT
00654 014000          14000
00655 000013          13
00656 024541          LDA 1,C60
00657 123000          ADD 1,0
00660 006056          JSR @.PRINT
00661 020537          LDA 0,COMMA
00662 006056          JSR @.PRINT
00663 000412          JMP PT2
00664 006060 JUMP:    JSR @.SHIFT
00665 014000          14000
00666 000013          13
00667 103000          ADD 0,0
00670 030507          LDA 2,ADR.1
00671 113000          ADD 0,2
00672 006050          JSR @.PRTXT
00673 020523          LDA 0,BLANK
00674 006056          JSR @.PRINT
00675 006060 PT2:     JSR @.SHIFT
00676 002000          2000
00677 000012          12
00700 101005          MOV 0,0,SNR
00701 000403          JMP .A2
00702 020517          LDA 0,AT
00703 006056          JSR @.PRINT
00704 006060 .A2:     JSR @.SHIFT
```

```
00705 001400           1400
00706 000010           10
00707 101005           MOV 0,0,SNR
00710 000430           JMP PAGE0
00711 126520           SUBZL 1,1
00712 106414           SUB# 0,1,SZR
00713 000510           JMP INDEX ;INDEX IS AC 2 OR 3
00714 020506           LDA 0,DOT ;INDEX IS PC
00715 006056           JSR @.PRINT
00716 004441           JSR FIX
00717 020477           LDA 0,BLANK
00720 006056           JSR @.PRINT
00721 000403           JMP TAG8
00722 000177 XMSK:     177
00723 000200 XMSK1:    200
00724 020057 TAG8:     LDA 0,INSTR
00725 024775           LDA 1,XMSK
00726 030775           LDA 2,XMSK1
00727 113404           AND 0,2,SZR
00730 004416           JSR T8.1
00731 107400           AND 0,1
00732 020051           LDA 0,PC
00733 123000           ADD 1,0
00734 126400           SUB 1,1
00735 006053           JSR @.OCTAL
00736 002457           JMP @RET
00737 000377           377
00740 020057 PAGE0:    LDA 0,INSTR
00741 024776           LDA 1,PAGE0-1
00742 123400           AND 1,0
00743 126400           SUB 1,1
00744 006053           JSR @.OCTAL
00745 002450           JMP @RET
00746 100400 T8.1:     NEG 0,0
00747 107400           AND 0,1
00750 124400           NEG 1,1
00751 001401           JMP 1,3
00752 000053 PLUS:     "+
00753 000055 MINUS:    "-
00754 000177 DMSK:     177
00755 000200 SMSK:     200
00756 000000           0
00757 054777 FIX:      STA 3,.-1 ;SUBRTN TO PRINT DISPLACEMENT
00760 020772           LDA 0,PLUS ;ASSUME PLUS
00761 030057           LDA 2,INSTR
00762 024773           LDA 1,SMSK ;GET SIGN MASK
00763 133414           AND# 1,2,SZR ;AND TEST SIGN
00764 004410           JSR F.2 ;DISPL IS NEGATIVE!
00765 006056           JSR @.PRINT
00766 141000           MOV 2,0
00767 024765           LDA 1,DMSK
00770 123400           AND 1,0
00771 126400           SUB 1,1
00772 006053           JSR @.OCTAL
00773 002763           JMP @FIX-1 ;RETURN
00774 150400 F.2:      NEG 2,2
00775 020756           LDA 0,MINUS
00776 001400           JMP 0,3
00777 001000 ADR.1:    .+1
01000 046512 .TXT *JM
```

```
01001 000120 P*

01002 051512 .TXT *JS
01003 000122 R*

01004 051511 .TXT *IS
01005 000132 Z*

01006 051504 .TXT *DS
01007 000132 Z*

01010 001011 ADR.2:   .+1
01011 042114 .TXT *LD
01012 000101 A*

01013 052123 .TXT *ST
01014 000101 A*

01015 000000 RET:     0
01016 000040 BLANK:   40
01017 000060 C60:     60
01020 000054 COMMA:   ",
01021 000100 AT:      "@
01022 000056 DOT:     ".
01023 004734 INDEX:   JSR FIX
01024 020774          LDA 0,COMMA
01025 006056          JSR @.PRINT
01026 006060          JSR @.SHIFT
01027 001400          1400
01030 000010          10
01031 024766          LDA 1,C60
01032 123000          ADD 1,0
01033 006056          JSR @.PRINT
01034 002761          JMP @RET
01035 006060 IO:      JSR @.SHIFT ;ITS AN IO TYPE
01036 003400          3400
01037 000010          10
01040 030470          LDA 2,ADR10
01041 103000          ADD 0,0
01042 113000          ADD 0,2
01043 006050          JSR @.PRTXT
01044 006060          JSR @.SHIFT
01045 003400          3400
01046 000010          10
01047 024520          LDA 1,C7
01050 106415          SUB# 0,1,SNR
01051 000410          JMP IO5
01052 006060          JSR @.SHIFT
01053 000300          300
01054 000006          6
01055 030474          LDA 2,ADR11
01056 113000          ADD 0,2
01057 006050          JSR @.PRTXT
01060 000410          JMP IO12
01061 006060 IO5:     JSR @.SHIFT
01062 000300          300
01063 000006          6
01064 030472          LDA 2,ADR12
01065 103000          ADD 0,0
01066 113000          ADD 0,2
```

```
01067 006050           JSR @.PRTXT
01070 020726 IO12:     LDA 0,BLANK
01071 006056           JSR @.PRINT
01072 006060           JSR @.SHIFT
01073 014000           14000
01074 000013           13
01075 126400           SUB 1,1
01076 006053           JSR @.OCTAL
01077 020721           LDA 0,COMMA
01100 006056           JSR @.PRINT
01101 006060           JSR @.SHIFT
01102 000077           77
01103 000000           0
01104 024776           LDA 1,.-2
01105 030477           LDA 2,ADR16
01106 106415           SUB# 0,1,SNR
01107 000417           JMP XCPU
01110 024460           LDA 1,C10
01111 030462           LDA 2,ADR15
01112 034457           LDA 3,C4
01113 054457           STA 3,CT4
01114 122415 IOLP:     SUB# 1,0,SNR
01115 000411           JMP XCPU
01116 125400           INC 1,1
01117 151400           INC 2,2
01120 151400           INC 2,2
01121 014451           DSZ CT4
01122 000772           JMP IOLP
01123 126400           SUB 1,1
01124 006053           JSR @.OCTAL
01125 002670           JMP @RET
01126 006050 XCPU:     JSR @.PRTXT
01127 002666           JMP @RET
01130 001131 ADR10:    .+1
01131 044516 .TXT *NI
01132 000117 O*

01133 044504 .TXT *DI
01134 000101 A*

01135 047504 .TXT *DO
01136 000101 A*

01137 044504 .TXT *DI
01140 000102 B*

01141 047504 .TXT *DO
01142 000102 B*

01143 044504 .TXT *DI
01144 000103 C*

01145 047504 .TXT *DO
01146 000103 C*

01147 045523 .TXT *SK
01150 000120 P*

01151 001152 ADR11:    .+1
01152 000000           0
```

```
01153 000123          "S
01154 000103          "C
01155 000120          "P
01156 001157 ADR12:   .+1
01157 047102 .TXT *BN
01160 000000 *

01161 055102 .TXT *B↑
01162 000000 *

01163 047104 .TXT *DN
01164 000000 *

01165 055104 .TXT *D↑
01166 000000 *

01167 000007 C7:      7
01170 000010 C10:     10
01171 000004 C4:      4
01172 000000 CT4:     0
01173 001174 ADR15:   .+1.
01174 052124 .TXT *TT
01175 000111 I*

01176 052124 .TXT *TT
01177 000117 O*

01200 052120 .TXT *PT
01201 000122 R*

01202 052120 .TXT *PT
01203 000120 P*

01204 001205 ADR16:   .+1
01205 050103 .TXT *CP
01206 000125 U*

01207 054467 ARITH:   STA 3,SV3 ;SUBRTN TO PROCESS ALC TYPES
01210 004543          JSR SHIFT
01211 003400          3400
01212 000010          10
01213 103000          ADD 0,0
01214 030574          LDA 2,ADR1
01215 113000          ADD 0,2
01216 004554          JSR PRTXT
01217 004534          JSR SHIFT
01220 000060          60
01221 000004          4
01222 030101          LDA 2,.ADR2
01223 113000          ADD 0,2
01224 004546          JSR PRTXT
01225 004526          JSR SHIFT
01226 000300          300
01227 000006          6
01230 030102          LDA 2,.ADR3
01231 113000          ADD 0,2
01232 004540          JSR PRTXT
01233 004520          JSR SHIFT
01234 000010          10
01235 000003          3
```

```
01332 060177           INTEN
01333 002747           JMP @PRINT-1
01334 060177 P.35:     INTEN
01335 000401           JMP .+1 ;GIVE TIME FOR INTERRUPT
01336 000401           JMP .+1
01337 060277           INTDS
01340 000763           JMP P.30 ;GO TRY NOW
01341 000012 C12:      12
01342 000015 C15:      15
01343 020777 CR:       LDA 0,C15 ;SUBRTN TO DO A CR LF
01344 054405           STA 3,SV32
01345 004736           JSR PRINT
01346 020773           LDA 0,C12
01347 004734           JSR PRINT
01350 002401           JMP @SV32
01351 000000 SV32:     0
01352 000000           0
01353 044777 SHIFT:    STA 1,.-1  ;JSR SHIFT
01354 020057           LDA 0,INSTR ;AND MASK
01355 025400           LDA 1,0,3  ;AMOUNT TO RIGHT SHIFT RESULT
01356 123400           AND 1,0    ;RETURN
01357 025401           LDA 1,1,3
01360 124405           NEG 1,1,SNR
01361 000404           JMP .+4
01362 101220           MOVZR 0,0
01363 125404           INC 1,1,SZR
01364 000776           JMP .-2
01365 024765           LDA 1,SHIFT-1
01366 001402           JMP 2,3
01367 177400 PMSK.:    177400
01370 000000 PSV1:     0
01371 000000 PSV3:     0
01372 044776 PRTXT:    STA 1,PSV1 ;SUBRTN TO PRINT A
01373 054776           STA 3,PSV3 ;TEXT STRING WHOSE
01374 024773           LDA 1,PMSK. ;ADDRESS IS IN AC2
01375 021000 PLOOP:    LDA 0,0,2
01376 101005           MOV 0,0,SNR
01377 000407           JMP P30
01400 004703           JSR PRINT
01401 123705           ANDS 1,0,SNR
01402 000404           JMP P30
01403 004700           JSR PRINT
01404 151400           INC 2,2
01405 000770           JMP PLOOP
01406 024762 P30:      LDA 1,PSV1
01407 002762           JMP @PSV3
01410 001411 ADR1:     .+1
01411 047503 .TXT *CO
01412 000115 M*

01413 042516 .TXT *NE
01414 000107 G*

01415 047515 .TXT *MO
01416 000126 V*

01417 047111 .TXT *IN
01420 000103 C*

01421 042101 .TXT *AD
```

```
01236 030103          LDA 2,.ADR4
01237 113000          ADD 0,2
01240 004532          JSR PRTXT
01241 020434          LDA 0,SPACE
01242 004441          JSR PRINT
01243 004510          JSR SHIFT
01244 060000          60000
01245 000015          15
01246 030104          LDA 2,.ADR5
01247 113000          ADD 0,2
01250 004522          JSR PRTXT
01251 020423          LDA 0,CMMA
01252 004431          JSR PRINT
01253 004500          JSR SHIFT
01254 014000          14000
01255 000013          13
01256 030570          LDA 2,ADR5
01257 113000          ADD 0,2
01260 004512          JSR PRTXT
01261 004472          JSR SHIFT
01262 000007          7
01263 000000          0
01264 103005          ADD 0,0,SNR
01265 000406          JMP END2
01266 030565          LDA 2,ADR6
01267 113000          ADD 0,2
01270 020404          LDA 0,CMMA
01271 004412          JSR PRINT
01272 004500          JSR PRTXT
01273 002403 END2:    JMP @SV3
01274 000054 CMMA:    ",
01275 000040 SPACE:   40
01276 000000 SV3:     0
01277 000000          0
01300 000000          0
01301 000000          0
01302 000000          0
01303 054777 PRINT:   STA 3,.-1
01304 040773          STA 0,PRINT-4
01305 044773          STA 1,PRINT-3
01306 050773          STA 2,PRINT-2
01307 101005          MOV 0,0,SNR
01310 001400          JMP 0,3
01311 060277          INTDS
01312 024100          LDA 1,OBUF+6 ;TTO ACTIVE FLAG
01313 125004          MOV 1,1,SZR
01314 000407          JMP P.30
01315 061111          DOAS 0,TTO
01316 126000          ADC 1,1
01317 044100          STA 1,OBUF+6 ;SET FLAG ON
01320 024760          LDA 1,PRINT-3
01321 060177          INTEN
01322 002760          JMP @PRINT-1
01323 006062 P.30:    JSR @INBUF
01324 000072          OBUF
01325 000407          JMP P.35 ;FULL!
01326 000401          JMP .+1 ;BECAME FULL
01327 020750          LDA 0,PRINT-4
01330 024750          LDA 1,PRINT-3
01331 030750          LDA 2,PRINT-2
```

```
01422 000103 C*

01423 052523 .TXT *SU
01424 000102 B*

01425 042101 .TXT *AD
01426 000104 D*

01427 047101 .TXT *AN
01430 000104 D*

01431 001432 ADR2:   .+1
01432 000000          0
01433 000132          "Z
01434 000117          "O
01435 000103          "C
01436 001437 ADR3:    .+1
01437 000000          0
01440 000114          "L
01441 000122          "R
01442 000123          "S
01443 001444 ADR4:    .+1
01444 000000          0
01445 000043          "#
01446 001447 ADR5:    .+1
01447 000060          "0
01450 000061          "1
01451 000062          "2
01452 000063          "3
01453 001454 ADR6:    .+1
01454 000000          0
01455 000000          0
01456 045523 .TXT *SK
01457 000120 P*

01460 055123 .TXT *SZ
01461 000103 C*

01462 047123 .TXT *SN
01463 000103 C*

01464 055123 .TXT *SZ
01465 000122 R*

01466 047123 .TXT *SN
01467 000122 R*

01470 042523 .TXT *SE
01471 000132 Z*

01472 041123 .TXT *SB
01473 000116 N*

01474 000000          0
01475 000006          6
01476 000060          60
01477 000000          0
01500 000000          0
01501 054777 OCTAL:   STA 3,.-1      ;A SUBRTN TO PRINT AC0
01502 044775          STA 1,OCTAL-2 ;IN OCTAL; SUPPRESS LEAD ZEROS
```

```
---
01503 024772          LDA 1,OCTAL-4 ;IF AC1=0
01504 044770          STA 1,OCTAL-5
01505 105000          MOV 0,1
01506 102400          SUB 0,0
01507 000406          JMP OLP.2
01510 102400 OLP.1:   SUB 0,0
01511 125120          MOVZL 1,1
01512 101100          MOVL 0,0
01513 125120          MOVZL 1,1
01514 101100          MOVL 0,0
01515 125120 OLP.2:   MOVZL 1,1
01516 101105          MOVL 0,0,SNR
01517 004414          JSR OC.5
01520 030756          LDA 2,OCTAL-3
01521 143000          ADD 2,0
01522 050755          STA 2,OCTAL-2
01523 006056          JSR @.PRINT
01524 014750 OC.2:    JSR OCTAL-5
01525 000763          JMP OLP.1
01526 020750          LDA 0,OCTAL-3
01527 024750          LDA 1,OCTAL-2
01530 125005          MOV 1,1,SNR
01531 006056          JSR @.PRINT
01532 002746          JMP @OCTAL-1
01533 030744 OC.5:    LDA 2,OCTAL-2
01534 151004          MOV 2,2,SZR
01535 001400          JMP 0,3
01536 000766          JMP OC.2
01537 040427 INTRT:   STA 0,SVAC
01540 044427          STA 1,SVAC+1
01541 050427          STA 2,SVAC+2
01542 054427          STA 3,SVAC+3
01543 101100          MOVL 0,0
01544 040426          STA 0,SVAC+4
01545 061477          INTA 0
01546 063710          SKPDZ TTI
01547 000425          JMP SVTTI
01550 063711          SKPDZ TTO
01551 000436          JMP SVTTO
01552 024421          LDA 1,FAKR
01553 123000          ADD 1,0
01554 040401          STA 0,.+1
01555 000000          0
01556 020414 DISMS:   LDA 0,SVAC+4
01557 101200          MOVR 0,0
01560 020406          LDA 0,SVAC
01561 024406          LDA 1,SVAC+1
01562 030406          LDA 2,SVAC+2
01563 034406          LDA 3,SVAC+3
01564 060177          INTEN
01565 002000          JMP @0
      000005 SVAC:    .BLK 5
01573 060200 FAKR:    NIOC 0
01574 060610 SVTTI:   DIAC 0,TTI
01575 126400          SUB 1,1
01576 044071          STA 1,IBUF+6
01577 006062          JSR @INBUF
01600 000063          IBUF
01601 063077          HALT
01602 000754          JMP DISMS ;BECAME FULL
```

```
01735 005015 HDR1:    .TXT *<15><12>
01736 040523 SA
01737 020116 N
01740 047101 AN
01741 047524 TO
01742 044516 NI
01743 020117 O
01744 047503 CO
01745 046114 LL
01746 043505 EG
01747 020105 E
01750 046105 EL
01751 041505 EC
01752 051124 TR
01753 047117 ON
01754 041511 IC
01755 020123 S
01756 042504 DE
01757 040520 PA
01760 052122 RT
01761 042515 ME
01762 052116 NT
01763 005015 <15><12>
01764 040523 SA
01765 020116 N
01766 047101 AN
01767 047524 TO
01770 044516 NI
01771 026117 O,
01772 052040  T
01773 054105 EX
01774 051501 AS
01775 020040
01776 034067 78
01777 030462 21
02000 006462 2<15>
02001 005012 <12><12>
02002 044504 DI
02003 026523 S-
02004 051501 AS
02005 042523 SE
02006 041115 MB
02007 042514 LE
02010 040440  A
02011 020116 N
02012 041101 AB
02013 047523 SO
02014 052514 LU
02015 042524 TE
02016 041040  B
02017 047111 IN
02020 051101 AR
02021 020131 Y
02022 040524 TA
02023 042520 PE
02024 005015 <15><12>
02025 000000 *

      000002 .END 2
```

```
01603 060110          NIOS TTI
01604 102000          ADC 0,0
01605 040071          STA 0,IBUF+6
01606 000750          JMP DISMS
01607 006061 SVTTO:   JSR @OTBUF
01610 000072          OBUF
01611 000407          JMP .+7 ;EMPTY!
01612 101005          MOV 0,0,SNR
01613 000774          JMP SVTTO
01614 061111          DOAS 0,TTO
01615 102000          ADC 0,0
01616 040100          STA 0,OBUF+6
01617 000737          JMP DISMS
01620 102400          SUB 0,0
01621 040100          STA 0,OBUF+6
01622 060211          NIOC TTO
01623 000733          JMP DISMS
01624 000000          0
01625 054777 INBF:    STA 3,.-1
01626 035400          LDA 3,0,3
01627 010775          ISZ INBF-1
01630 025400          LDA 1,0,3 ;GET CHAR COUNT
01631 031401          LDA 2,1,3 ;AND MAX CT
01632 132415          SUB# 1,2,SNR
01633 002771          JMP @INBF-1 ;BUFFER IS FULL!
01634 010770          ISZ INBF-1
01635 125400          INC 1,1
01636 045400          STA 1,0,3 ;UPDATE CHAR CT
01637 132414          SUB# 1,2,SZR ;WILL BFR BECOME FULL?
01640 010764          ISZ INBF-1 ;NO
01641 043402          STA 0,@2,3 ;PUT CHAR IN BUFFER
01642 011402          ISZ 2,3 ;UPDATE CHAR POINTER
01643 025402          LDA 1,2,3 ;AND GRAB IT
01644 031405          LDA 2,5,3 ;ALSO RESET TEST
01645 132414          SUB# 1,2,SZR ;AND TEST
01646 002756          JMP @INBF-1 ;ALL OK
01647 025404          LDA 1,4,3 ;GET RESET POINTER
01650 045402          STA 1,2,3
01651 002753          JMP @INBF-1
01652 000000          0
01653 054777 OTBF:    STA 3,.-1
01654 035400          LDA 3,0,3
01655 010775          ISZ OTBF-1
01656 021400          LDA 0,0,3 ;GET CHAR CT
01657 101005          MOV 0,0,SNR
01660 002772          JMP @OTBF-1 ;EMPTY - RETURN
01661 010771          ISZ OTBF-1
01662 011403          ISZ 3,3
01663 031403          LDA 2,3,3 ;OUTPUT CHAR POINTER
01664 021377          LDA 0,-1,2 ;PULL FROM BUFFER
01665 015400          DSZ 0,3 ;ADJUST CHAR COUNT
01666 000401          JMP .+1 ;NO TEST
01667 025405          LDA 1,5,3 ;RESET TEST
01670 132414          SUB# 1,2,SZR
01671 002761          JMP @OTBF-1 ;NORMAL RETURN
01672 025404          LDA 1,4,3 ;RESET POINTER
01673 045403          STA 1,3,3 ;OUTPUT POINTER
01674 002756          JMP @OTBF-1
      000020 AREA1:   .BLK 20
      000020 AREA2:   .BLK 20
```

```
---
LOOP     000502
MINUS    000753
MSK8     000530
OBUF     000072
OCTAL    001501
OC.2     001524
OC.5     001533
OLP.1    001510
OLP.2    001515
OTBF     001653
OTBUF    000061
P30      001406
PAGE0    000740
PC       000051
PLOOP    001375
PLUS     000752
PMSK.    001367
PRINT    001303
PRTXT    001372
PSV1     001370
PSV3     001371
PT2      000675
P.30     001323
P.35     001334
RD2      000537
RD.6     000455
READ     000440
RET      001015
RPT      000422
RSTC     000434
SHIFT    001353
SMSK     000755
SPACE    001275
START    000400
SV3      001276
SV32     001351
SVAC     001566
SVTTI    001574
SVTTO    001607
T8.1     000746
TAG8     000724
TEMP     000546
XCPU     001126
XMSK     000722
XMSK1    000723
XXX      000520
XXX2     000525
.A2      000704
.ADR2    000101
.ADR3    000102
.ADR4    000103
.ADR5    000104
.ARIT    000054
.CR      000052
.LONG    000055
.OCTA    000053
.PRIN    000056
.PRTX    000050
.SHIF    000060

---
```

```
---
ADR1     001410
ADR10    001130
ADR11    001151
ADR12    001156
ADR15    001173
ADR16    001204
ADR2     001431
ADR3     001436
ADR4     001443
ADR5     001446
ADR6     001453
ADR7     000630
ADR.1    000777
ADR.2    001010
ADR.5    000616
ADR.6    000623
ADX3     000547
AREA1    001675
AREA2    001715
ARITH    001207
AT       001021
BLANK    001016
BLKS     000532
C10      001170
C12      001341
C15      001342
C4       001171
C40      000527
C60      001017
C7       001167
CMMA     001274
COMMA    001020
COUNT    000545
CR       001343
CT4      001172
DATA     000471
DISMS    001556
DMSK     000754
DOT      001022
END      000571
END2     001273
END3     000612
ERROR    000553
FAKE     001573
FIX      000757
F.2      000774
HDR      000105
HDR1     001735
IBUF     000063
INBF     001625
INBUF    000062
INDEX    001023
INSTR    000057
INTRT    001537
IO       001035
IO12     001070
IO5      001061
IOLP     001114
JUMP     000664
LONG     000635
```

```
MODE:1

                        .END

MODE:3

---

          000106        .LOC 106
   00106  020100        LDA 0,OBUF+6 ;WAIT FOR IO TO END
   00107  101004        MOV 0,0,SZR
   00110  000106        JMP .-2
   00111  063077        HALT
   00112  000002        JMP 2
          000610        .LOC 610
   00610  000106        JMP 106
          000614        .LOC 614
   00614  000106        JMP 106
          000072 OBUF=72

                        .END

---
OBUF      000072

---

MODE:
```

Correction to allow TTO to complete when END block is read.

included in object tape

START AT 2 OR 3

USERS GROUP LIBRARY
PROGRAM #7 DIS-ASSEMBLER
DATE SUBMITTED: FEBRUARY, 1971
SAN ANTONIO COLLEGE

```
      000001            .LOC 1
00001 001537            INTRT
00002 002004            JMP @4
00003 002004            JMP @4
00004 000400            400
      000050            .LOC 50
00050 001372 .PRTXT:    PRTXT
00051 000000 PC:        0 ;PSEUDO PROGRAM COUNTER
00052 001343 .CR:       CR
00053 001501 .OCTAL:    OCTAL
00054 001207 .ARITH:    ARITH
00055 000635 .LONG:     LONG
00056 001303 .PRINT:    PRINT
00057 000000 INSTR:     0  ;THE CURRENT INSTRUCTION
00060 001353 .SHIFT:    SHIFT
00061 001653 OTBUF:     OTBF
00062 001625 INBUF:     INBF
00063 000000 IBUF:      0       ;CHARACTER COUNT
00064 000020            20      ;MAX COUNT
00065 001675            AREA1   ;INPT PTR-VERY NEXT CHAR
00066 001675            AREA1   ;OTPT PTR-VERY NEXT CHAR
00067 001675            AREA1   ;RESET PTR
00070 001715            AREA1+20  ;RESET TEST
00071 000000            0         ;ACTIVE FLAG - TTI
00072 000000 OBUF:      0
00073 000020            20
00074 001715            AREA2
00075 001715            AREA2
00076 001715            AREA2
00077 001735            AREA2+20
00100 000000            0
00101 001432 .ADR2:     ADR2+1
00102 001437 .ADR3:     ADR3+1
00103 001444 .ADR4:     ADR4+1
00104 001447 .ADR5:     ADR5+1
00105 001735 HDR:       HDR1
      000400            .LOC 400
00400 060277 START:     INTDS
00401 062677            IORST
00402 060110            NIOS TTI
00403 102400            SUB 0,0
00404 040063            STA 0,IBUF
00405 040072            STA 0,OBUF
00406 040100            STA 0,OBUF+6
00407 020067            LDA 0,IBUF+4
00410 040065            STA 0,IBUF+2
00411 040066            STA 0,IBUF+3
00412 020076            LDA 0,OBUF+4
00413 040074            STA 0,OBUF+2
00414 040075            STA 0,OBUF+3
00415 102000            ADC 0,0
00416 040071            STA 0,IBUF+6
00417 060177            INTEN
00420 030105            LDA 2,HDR
00421 006050            JSR @.PRTXT
00422 004416 RPT:       JSR READ
00423 101005            MOV 0,0,SNR
00424 000776            JMP .-2
```

```
00425 024503          LDA 1,MSK8
00426 107404          AND 0,1,SZR
00427 000442          JMP DATA ;ITS A DATA BLOCK
00430 126520          SUBZL 1,1
00431 106405          SUB 0,1,SNR
00432 000537          JMP END ;ITS THE END BLOCK
00433 000520          JMP ERROR ;ITS AN ERROR BLOCK
00434 000007 RSTC:    7
00435 000000          0
00436 000000          0
00437 000000          0
00440 054777 READ:    STA 3,.-1
00441 050775          STA 2,READ-2
00442 044773          STA 1,READ-3
00443 020071          LDA 0,IBUF+6  ;GET TTI ACTIVE FLAG
00444 101004          MOV 0,0,SZR
00445 000410          JMP RD.6
00446 020063          LDA 0,IBUF  ;CHAR COUNT
00447 024765          LDA 1,RSTC ;RESTART CONSTANT
00450 106512          SUBL# 0,1,SZC
00451 000404          JMP RD.6  ;DONT START TTI
00452 060110          NIOS TTI
00453 102000          ADC 0,0
00454 040071          STA 0,IBUF+6 ;SET FLAG ON
00455 060277 RD.6:    INTDS
00456 000061          JSR @OTBUF
00457 000063          IBUF
00460 000405          JMP .+5 ;EMPTY!
00461 024754          LDA 1,READ-3
00462 034754          LDA 2,READ-2
00463 060177          INTEN
00464 002752          JMP @READ-2
00465 060177          INTEN
00466 000401          JMP .+1 ;GIVE TIME FOR INTERRUPT
00467 000401          JMP .+1
00470 000765          JMP RD.6
00471 125400 DATA:    MOVS 0,1
00472 004746          JSR READ
00473 107000          ADDS 0,1
00474 044451          STA 1,COUNT ;THE WORD COUNT
00475 000052          JSR @.CR
00476 000052          JSR @.CR
00477 004440          JSR RD2
00500 040051          STA 0,PC ;PSEUDO PROGRAM COUNTER
00501 004436          JSR RD2 ;IGNORE CHECK SUM
00502 004435 LOOP:    JSR RD2 ;GO GET INSTRUCTION
00503 040057          STA 0,INSTR ;AND STORE IT
00504 020051          LDA 0,PC
00505 126400          SUB 1,1 ;SUPPRESS LEAD ZEROS
00506 006053          JSR @.OCTAL ;AS CORE LOCN PRINTS
00507 004423          JSR BLKS ;LEAVE TWO SPACES
00510 020057          LDA 0,INSTR
00511 126000          ADC 1,1
00512 006053          JSR @.OCTAL ;PRINT THE INSTR IN OCTAL
00513 004417          JSR BLKS
00514 020057          LDA 0,INSTR
00515 101113          MOVL# 0,0,SNC
00516 000407          JMP XXX2 ;NOT AN ALC TYPE
00517 006054          JSR @.ARITH ;IT IS AN ALC
00520 006052 XXX:     JSR @.CR
```

```
00521 010051            ISZ PC ;BUMP PROGRAM COUNT
00522 010423            ISZ COUNT ;BUMP WORD COUNT
00523 000757            JMP LOOP ;BLOCK NOT FINISHED
00524 000676            JMP RPT ;GO FIND NEXT BLOCK
00525 006055 XXX2:      JSR @.LONG ;GO PROCESS NON ALC TYPE
00526 000772            JMP XXX
00527 000040 C40:       40
00530 000200 MSK0:      200
00531 000000            0
00532 054777 BLKS:      STA 3,.-1
00533 020774            LDA 0,C40
00534 006056            JSR @.PRINT
00535 006056            JSR @.PRINT
00536 002773            JMP @BLKS-1
00537 054772 RD2:       STA 3,BLKS-1 ;SUBRTN TO READ TWO INPUTS
00540 004700            JSR READ     ;FROM TTI AND BUILD INSTR WORD
00541 105300            MOVS 0,1
00542 004676            JSR READ
00543 123300            ADDS 1,0
00544 002765            JMP @BLKS-1
00545 000000 COUNT:     0
00546 000000 TEMP:      0
00547 000550 ADX3:      .+1
00550 051105 .TXT *ER
00551 047522 RO
00552 000122 R*

00553 040773 ERROR:     STA 0,TEMP ;SUBRTN TO PROCESS AN
00554 006052            JSR @.CR   ;ERROR BLOCK
00555 006052            JSR @.CR
00556 030771            LDA 2,ADX3
00557 006050            JSR @.PRTXT
00560 006052            JSR @.CR
00561 020765            LDA 0,TEMP
00562 006056            JSR @.PRINT
00563 004655            JSR READ
00564 101004            MOV 0,0,SZR
00565 000775            JMP .-3
00566 006052            JSR @.CR
00567 006052            JSR @.CR
00570 000632            JMP RPT
00571 006052 END:       JSR @.CR  ;SUBRTN TO PROCESS END BLOCKS
00572 006052            JSR @.CR  ;AND HALT THE PROGRAM
00573 030423            LDA 2,ADR.5
00574 006050            JSR @.PRTXT
00575 004643            JSR READ
00576 004741            JSR RD2
00577 101112            MOVL# 0,0,SZC
00600 000412            JMP END3
00601 040745            STA 0,TEMP
00602 030421            LDA 2,ADR.6
00603 006050            JSR @.PRTXT
00604 020742            LDA 0,TEMP
00605 126400            SUB 1,1
00606 006053            JSR @.OCTAL
00607 004730            JSR RD2
00610 063077            HALT          See correction
00611 000777            JMP .-1
00612 030416 END3:      LDA 2,ADR7
00613 006050            JSR @.PRTXT
```

```
00614 063077            HALT        See correction
00615 000777            JMP .-1
00616 000617 ADR.5:     .+1
00617 047105 .TXT *EN
00620 020104 D
00621 020040
00622 000000 *

00623 000624 ADR.6:     .+1
00624 047507 .TXT *GO
00625 052040  T
00626 020117 O
00627 000040  *

00630 000631 ADR7:      .+1
00631 040510 .TXT *HA
00632 052114 LT
00633 000040  *

00634 001035            IO
00635 054560 LONG:      STA 3,RET ;SUBRTN TO PROCESS NON ALC TYPES
00636 006060            JSR @.SHIFT
00637 060000            60000
00640 000015            15
00641 101005            MOV 0,0,SNR
00642 000422            JMP JUMP ;MEM REF W/O AC TYPE
00643 101226            MOVZR 0,0,SEZ
00644 000770            JMP @LONG-1 ;ITS AN IO TYPE
00645 030543            LDA 2,ADR.2
00646 103000            ADD 0,0
00647 113000            ADD 0,2
00650 006050            JSR @.PRTXT
00651 020545            LDA 0,BLANK
00652 006056            JSR @.PRINT
00653 006060            JSR @.SHIFT
00654 014000            14000
00655 000013            13
00656 024541            LDA 1,C60
00657 123000            ADD 1,0
00660 006056            JSR @.PRINT
00661 020537            LDA 0,COMMA
00662 006056            JSR @.PRINT
00663 000412            JMP PT2
00664 006060 JUMP:      JSR @.SHIFT
00665 014000            14000
00666 000013            13
00667 103000            ADD 0,0
00670 030507            LDA 2,ADR.1
00671 113000            ADD 0,2
00672 006050            JSR @.PRTXT
00673 020523            LDA 0,BLANK
00674 006056            JSR @.PRINT
00675 006060 PT2:       JSR @.SHIFT
00676 002000            2000
00677 000012            12
00700 101005            MOV 0,0,SNR
00701 000403            JMP .A2
00702 020517            LDA 0,AT
00703 006056            JSR @.PRINT
00704 006060 .A2:       JSR @.SHIFT
```

```
00705 001400          1400
00706 000010          10
00707 101005          MOV 0,0,SNR
00710 000430          JMP PAGE0
00711 126520          SUBEL 1,1
00712 106414          SUB# 0,1,SZR
00713 000510          JMP INDEX ;INDEX IS AC 2 OR 3
00714 020506          LDA 0,DOT ;INDEX IS PC
00715 006056          JSR @.PRINT
00716 004441          JSR FIX
00717 020477          LDA 0,BLANK
00720 006056          JSR @.PRINT
00721 000403          JMP TAG8
00722 000177 XMSK:    177
00723 000200 XMSK1:   200
00724 020057 TAG8:    LDA 0,INSTR
00725 024775          LDA 1,XMSK
00726 030775          LDA 2,XMSK1
00727 113404          AND 0,2,SZR
00730 004416          JSR T8.1
00731 107400          AND 0,1
00732 020051          LDA 0,PC
00733 123000          ADD 1,0
00734 126400          SUB 1,1
00735 006053          JSR @.OCTAL
00736 002457          JMP @RET
00737 000377          377
00740 020057 PAGE0:   LDA 0,INSTR
00741 024776          LDA 1,PAGE0-1
00742 123400          AND 1,0
00743 126400          SUB 1,1
00744 006053          JSR @.OCTAL
00745 002450          JMP @RET
00746 100400 T8.1:    NEG 0,0
00747 107400          AND 0,1
00750 124400          NEG 1,1
00751 001401          JMP 1,3
00752 000053 PLUS:    "+
00753 000055 MINUS:   "-
00754 000177 DMSK:    177
00755 000200 SMSK:    200
00756 000000          0
00757 054777 FIX:     STA 3,.-1 ;SUBRTN TO PRINT DISPLACEMENT
00760 020772          LDA 0,PLUS ;ASSUME PLUS
00761 030057          LDA 2,INSTR
00762 024773          LDA 1,SMSK ;GET SIGN MASK
00763 133414          AND# 1,2,SZR ;AND TEST SIGN
00764 004410          JSR F.2 ;DISPL IS NEGATIVE!
00765 006056          JSR @.PRINT
00766 141000          MOV 2,0
00767 024765          LDA 1,DMSK
00770 123400          AND 1,0
00771 126400          SUB 1,1
00772 006053          JSR @.OCTAL
00773 002763          JMP @FIX-1 ;RETURN
00774 150400 F.2:     NEG 2,2
00775 020756          LDA 0,MINUS
00776 001400          JMP 0,3
00777 001000 ADR.1:   .+1
01000 046512 .TXT *JM
```

```
01001 000120 P*

01002 051512 .TXT *JS
01003 000122 R*

01004 051511 .TXT *IS
01005 000132 Z*

01006 051504 .TXT *DS
01007 000132 Z*

01010 001011 ADR.2:  .+1
01011 042114 .TXT *LD
01012 000101 A*

01013 052123 .TXT *ST
01014 000101 A*

01015 000000 RET:    0
01016 000040 BLANK:  40
01017 000060 C60:    60
01020 000054 COMMA:  ",
01021 000100 AT:     "@
01022 000056 DOT:    ".
01023 004734 INDEX:  JSR FIX
01024 020774         LDA 0,COMMA
01025 006056         JSR @.PRINT
01026 006060         JSR @.SHIFT
01027 001400         1400
01030 000010         10
01031 024766         LDA 1,C60
01032 123000         ADD 1,0
01033 006056         JSR @.PRINT
01034 002761         JMP @RET
01035 006060 IO:     JSR @.SHIFT ;ITS AN IO TYPE
01036 003400         3400
01037 000010         10
01040 030470         LDA 2,ADR10
01041 103000         ADD 0,0
01042 113000         ADD 0,2
01043 006050         JSR @.PRTXT
01044 006060         JSR @.SHIFT
01045 003400         3400
01046 000010         10
01047 024520         LDA 1,C7
01050 106415         SUB# 0,1,SNR
01051 000410         JMP IO5
01052 006060         JSR @.SHIFT
01053 000300         300
01054 000006         6
01055 030474         LDA 2,ADR11
01056 113000         ADD 0,2
01057 006050         JSR @.PRTXT
01060 000410         JMP IO12
01061 006060 IO5:    JSR @.SHIFT
01062 000300         300
01063 000006         6
01064 030472         LDA 2,ADR12
01065 103000         ADD 0,0
01066 113000         ADD 0,2
```

```
01067 006050           JSR 0,.PRTXT
01070 020726 IO12:     LDA 0,BLANK
01071 006056           JSR @.PRINT
01072 006060           JSR @.SHIFT
01073 014000           14000
01074 000013           13
01075 126400           SUB 1,1
01076 006053           JSR @.OCTAL
01077 020721           LDA 0,COMMA
01100 006056           JSR @.PRINT
01101 006060           JSR @.SHIFT
01102 000077           77
01103 000000           0
01104 024776           LDA 1,.-2
01105 030477           LDA 2,ADR16
01106 106415           SUB# 0,1,SNR
01107 000417           JMP XCPU
01110 024460           LDA 1,C10
01111 030462           LDA 2,ADR15
01112 034457           LDA 3,C4
01113 054457           STA 3,CT4
01114 122415 IOLP:     SUB# 1,0,SNR
01115 000411           JMP XCPU
01116 125400           INC 1,1
01117 151400           INC 2,2
01120 151400           INC 2,2
01121 014451           DSZ CT4
01122 000772           JMP IOLP
01123 126400           SUB 1,1
01124 006053           JSR @.OCTAL
01125 002670           JMP @RET
01126 006050 XCPU:     JSR @.PRTXT
01127 002666           JMP @RET
01130 001131 ADR10:    .+1
01131 044516 .TXT *NI
01132 000117 O*

01133 044504 .TXT *DI
01134 000101 A*

01135 047504 .TXT *DO
01136 000101 A*

01137 044504 .TXT *DI
01140 000102 B*

01141 047504 .TXT *DO
01142 000102 B*

01143 044504 .TXT *DI
01144 000103 C*

01145 047504 .TXT *DO
01146 000103 C*

01147 045523 .TXT *SK
01150 000120 P*

01151 001152 ADR11:    .+1
01152 000000           0
```

```
01153 000123          "S
01154 000103          "C
01155 000120          "P
01156 001157 ADR12:   .+1
01157 047102 .TXT *BN
01160 000000 *

01161 055102 .TXT *B#
01162 000000 *

01163 047104 .TXT *DN
01164 000000 *

01165 055104 .TXT *D#
01166 000000 *

01167 000007 C7:      7
01170 000010 C10:     10
01171 000004 C4:      4
01172 000000 CT4:     0
01173 001174 ADR15:   .+1.
01174 052124 .TXT *TT
01175 000111 I*

01176 052124 .TXT *TT
01177 000117 O*

01200 052120 .TXT *PT
01201 000122 R*

01202 052120 .TXT *PT
01203 000120 P*

01204 001205 ADR16:   .+1
01205 050103 .TXT *CP
01206 000125 U*

01207 054467 ARITH:   STA 3,SV3 ;SUBRTN TO PROCESS ALC TYPES
01210 004543          JSR SHIFT
01211 003400          3400
01212 000010          10
01213 103000          ADD 0,0
01214 030574          LDA 2,ADR1
01215 113000          ADD 0,2
01216 004554          JSR PRTXT
01217 004534          JSR SHIFT
01220 000060          60
01221 000004          4
01222 030101          LDA 2,.ADR2
01223 113000          ADD 0,2
01224 004546          JSR PRTXT
01225 004526          JSR SHIFT
01226 000300          300
01227 000006          6
01230 030102          LDA 2,.ADR3
01231 113000          ADD 0,2
01232 004540          JSR PRTXT
01233 004520          JSR SHIFT
01234 000010          10
01235 000003          3
```

```
---
01236 030103          LDA 2,.ADR4
01237 113000          ADD 0,2
01240 004532          JSR PRTXT
01241 020434          LDA 0,SPACE
01242 004441          JSR PRINT
01243 004516          JSR SHIFT
01244 060000          60000
01245 000015          15
01246 030104          LDA 2,.ADR5
01247 113000          ADD 0,2
01250 004522          JSR PRTXT
01251 020423          LDA 0,CMMA
01252 004431          JSR PRINT
01253 004500          JSR SHIFT
01254 014000          14000
01255 000013          13
01256 030570          LDA 2,ADR5
01257 113000          ADD 0,2
01260 004512          JSR PRTXT
01261 004472          JSR SHIFT
01262 000007          7
01263 000000          0
01264 103005          ADD 0,0,SNR
01265 000406          JMP END2
01266 030565          LDA 2,ADR6
01267 113000          ADD 0,2
01270 020404          LDA 0,CMMA
01271 004412          JSR PRINT
01272 004500          JSR PRTXT
01273 002403 END2:    JMP @SV3
01274 000054 CMMA:    ",
01275 000040 SPACE:   40
01276 000000 SV3:     0
01277 000000          0
01300 000000          0
01301 000000          0
01302 000000          0
01303 054777 PRINT:   STA 3,.-1
01304 040773          STA 0,PRINT-4
01305 044773          STA 1,PRINT-3
01306 050773          STA 2,PRINT-2
01307 101005          MOV 0,0,SNR
01310 001400          JMP 0,3
01311 060277          INTDS
01312 024100          LDA 1,OBUF+6 ;TTO ACTIVE FLAG
01313 125004          MOV 1,1,SZR
01314 000407          JMP P.30
01315 061111          DOAS 0,TTO
01316 126000          ADC 1,1
01317 044100          STA 1,OBUF+6 ;SET FLAG ON
01320 024760          LDA 1,PRINT-3
01321 060177          INTEN
01322 002760          JMP @PRINT-1
01323 006062 P.30:    JSR @INBUF
01324 000072          OBUF
01325 000407          JMP P.35 ;FULL!
01326 000401          JMP .+1 ;BECAME FULL
01327 020750          LDA 0,PRINT-4
01330 024750          LDA 1,PRINT-3
01331 030750          LDA 2,PRINT-2
```

```
01332 060177          INTEN
01333 002747          JMP @PRINT-1
01334 060177 P.35:    INTEN
01335 000401          JMP .+1 ;GIVE TIME FOR INTERRUPT
01336 000401          JMP .+1
01337 060277          INTDS
01340 000763          JMP P.30 ;GO TRY NOW
01341 000012 C12:     12
01342 000015 C15:     15
01343 020777 CR:      LDA 0,C15 ;SUBRTN TO DO A CR LF
01344 054405          STA 3,SV32
01345 004736          JSR PRINT
01346 020773          LDA 0,C12
01347 004734          JSR PRINT
01350 002401          JMP @SV32
01351 000000 SV32:    0
01352 000000          0
01353 044777 SHIFT:   STA 1,.-1  ;JSR SHIFT
01354 020057          LDA 0,INSTR ;AND MASK
01355 025400          LDA 1,0,3  ;AMOUNT TO RIGHT SHIFT RESULT
01356 123400          AND 1,0    ;RETURN
01357 025401          LDA 1,1,3
01360 124405          NEG 1,1,SNR
01361 000404          JMP .+4
01362 101220          MOVZR 0,0
01363 125404          INC 1,1,SZR
01364 000776          JMP .-2
01365 024765          LDA 1,SHIFT-1
01366 001402          JMP 2,3
01367 177400 PMSK.:   177400
01370 000000 PSV1:    0
01371 000000 PSV3:    0
01372 044776 PRTXT:   STA 1,PSV1 ;SUBRTN TO PRINT A
01373 054776          STA 3,PSV3 ;TEXT STRING WHOSE
01374 024773          LDA 1,PMSK. ;ADDRESS IS IN AC2
01375 021000 PLOOP:   LDA 0,0,2
01376 101005          MOV 0,0,SNR
01377 000407          JMP P30
01400 004703          JSR PRINT
01401 123705          ANDS 1,0,SNR
01402 000404          JMP P30
01403 004700          JSR PRINT
01404 151400          INC 2,2
01405 000770          JMP PLOOP
01406 024762 P30:     LDA 1,PSV1
01407 002762          JMP @PSV3
01410 001411 ADR1:    .+1
01411 047503 .TXT *CO
01412 000115 M*

01413 042516 .TXT *NE
01414 000107 G*

01415 047515 .TXT *MO
01416 000126 V*

01417 047111 .TXT *IN
01420 000103 C*

01421 042101 .TXT *AD
```

```
01422 000103 C*

01423 052523 .TXT *SU
01424 000102 B*

01425 042101 .TXT *AD
01426 000104 D*

01427 047101 .TXT *AN
01430 000104 D*

01431 001432 ADR2:    .+1
01432 000000          0
01433 000132          "Z
01434 000117          "O
01435 000103          "C
01436 001437 ADR3:    .+1
01437 000000          0
01440 000114          "L
01441 000122          "R
01442 000123          "S
01443 001444 ADR4:    .+1
01444 000000          0
01445 000043          "#
01446 001447 ADR5:    .+1
01447 000060          "0
01450 000061          "1
01451 000062          "2
01452 000063          "3
01453 001454 ADR6:    .+1
01454 000000          0
01455 000000          0
01456 045523 .TXT *SK
01457 000120 P*

01460 055123 .TXT *SZ
01461 000103 C*

01462 047123 .TXT *SN
01463 000103 C*

01464 055123 .TXT *SZ
01465 000122 R*

01466 047123 .TXT *SN
01467 000122 R*

01470 042523 .TXT *SE
01471 000132 Z*

01472 041123 .TXT *SB
01473 000116 N*

01474 000000          0
01475 000006          6
01476 000060          60
01477 000000          0
01500 000000          0
01501 054777 OCTAL:   STA 3,.-1      ;A SUBRTN TO PRINT AC0
01502 044775          STA 1,OCTAL-2 ;IN OCTAL; SUPPRESS LEAD ZEROS
```

```
01503 024772          LDA 1,OCTAL-4 ;IF AC1=0
01504 044770          STA 1,OCTAL-5
01505 105000          MOV 0,1
01506 102400          SUB 0,0
01507 000406          JMP OLP.2
01510 102400 OLP.1:   SUB 0,0
01511 125120          MOVZL 1,1
01512 101100          MOVL 0,0
01513 125120          MOVZL 1,1
01514 101100          MOVL 0,0
01515 125120 OLP.2:   MOVZL 1,1
01516 101105          MOVL 0,0,SNR
01517 004414          JSR OC.5
01520 030756          LDA 2,OCTAL-3
01521 143000          ADD 2,0
01522 050755          STA 2,OCTAL-2
01523 006056          JSR @.PRINT
01524 014750 OC.2:    DSZ OCTAL-5
01525 000763          JMP OLP.1
01526 020750          LDA 0,OCTAL-3
01527 024750          LDA 1,OCTAL-2
01530 125005          MOV 1,1,SNR
01531 006056          JSR @.PRINT
01532 002746          JMP @OCTAL-1
01533 030744 OC.5:    LDA 2,OCTAL-2
01534 151004          MOV 2,2,SZR
01535 001400          JMP 0,3
01536 000766          JMP OC.2
01537 040427 INTRT:   STA 0,SVAC
01540 044427          STA 1,SVAC+1
01541 050427          STA 2,SVAC+2
01542 054427          STA 3,SVAC+3
01543 101100          MOVL 0,0
01544 040426          STA 0,SVAC+4
01545 061477          INTA 0
01546 063710          SKPDZ TTI
01547 000425          JMP SVTTI
01550 063711          SKPDZ TTO
01551 000436          JMP SVTTO
01552 024421          LDA 1,FAKR
01553 123000          ADD 1,0
01554 040401          STA 0,.+1
01555 000000          0
01556 020414 DISMS:   LDA 0,SVAC+4
01557 101200          MOVR 0,0
01560 020406          LDA 0,SVAC
01561 024406          LDA 1,SVAC+1
01562 030406          LDA 2,SVAC+2
01563 034406          LDA 3,SVAC+3
01564 060177          INTEN
01565 002000          JMP @0
      000005 SVAC:    .BLK 5
01573 060200 FAKR:    NIOC 0
01574 060610 SVTTI:   DIAC 0,TTI
01575 126400          SUB 1,1
01576 044071          STA 1,IBUF+6
01577 006062          JSR @INBUF
01600 000063          IBUF
01601 063077          HALT
01602 000754          JMP DISMS ;BECAME FULL
```

```
01603 060110          NIOS TTI
01604 102000          ADC 0,0
01605 040071          STA 0,IBUF+6
01606 000750          JMP DISMS
01607 006061 SVTTO:   JSR @OTBUF
01610 000072          OBUF
01611 000407          JMP .+7 ;EMPTY!
01612 101005          MOV 0,0,SNR
01613 000774          JMP SVTTO
01614 061111          DOAS 0,TTO
01615 102000          ADC 0,0
01616 040100          STA 0,OBUF+6
01617 000737          JMP DISMS
01620 102400          SUB 0,0
01621 040100          STA 0,OBUF+6
01622 060211          NIOC TTO
01623 000733          JMP DISMS
01624 000000          0
01625 054777 INBF:    STA 3,.-1
01626 035400          LDA 3,0,3
01627 010775          ISZ INBF-1
01630 025400          LDA 1,0,3 ;GET CHAR COUNT
01631 031401          LDA 2,1,3 ;AND MAX CT
01632 132415          SUB# 1,2,SNR
01633 002771          JMP @INBF-1 ;BUFFER IS FULL!
01634 010770          ISZ INBF-1
01635 125400          INC 1,1
01636 045400          STA 1,0,3 ;UPDATE CHAR CT
01637 132414          SUB# 1,2,SZR ;WILL BFR BECOME FULL?
01640 010764          ISZ INBF-1 ;NO
01641 043402          STA 0,@2,3 ;PUT CHAR IN BUFFER
01642 011402          ISZ 2,3 ;UPDATE CHAR POINTER
01643 025402          LDA 1,2,3 ;AND GRAB IT
01644 031405          LDA 2,5,3 ;ALSO RESET TEST
01645 132414          SUB# 1,2,SZR ;AND TEST
01646 002756          JMP @INBF-1 ;ALL OK
01647 025404          LDA 1,4,3 ;GET RESET POINTER
01650 045402          STA 1,2,3
01651 002753          JMP @INBF-1
01652 000000          0
01653 054777 OTBF:    STA 3,.-1
01654 035400          LDA 3,0,3
01655 010775          ISZ OTBF-1
01656 021400          LDA 0,0,3 ;GET CHAR CT
01657 101005          MOV 0,0,SNR
01660 002772          JMP @OTBF-1 ;EMPTY - RETURN
01661 010771          ISZ OTBF-1
01662 011403          ISZ 3,3
01663 031403          LDA 2,3,3 ;OUTPUT CHAR POINTER
01664 021377          LDA 0,-1,2 ;PULL FROM BUFFER
01665 015400          DSZ 0,3 ;ADJUST CHAR COUNT
01666 000401          JMP .+1 ;NO TEST
01667 025405          LDA 1,5,3 ;RESET TEST
01670 132414          SUB# 1,2,SZR
01671 002761          JMP @OTBF-1 ;NORMAL RETURN
01672 025404          LDA 1,4,3 ;RESET POINTER
01673 045403          STA 1,3,3 ;OUTPUT POINTER
01674 002756          JMP @OTBF-1
      000020 AREA1:   .BLK 20
      000020 AREA2:   .BLK 20
```

```
01735 005015 HDR1:    .TXT *<15><12>
01736 040523 SA
01737 020116 N
01740 047101 AN
01741 047524 TO
01742 044516 NI
01743 020117 O
01744 047503 CO
01745 046114 LL
01746 043505 EG
01747 020105 E
01750 046105 EL
01751 041505 EC
01752 051124 TR
01753 047117 ON
01754 041511 IC
01755 020123 S
01756 042504 DE
01757 040520 PA
01760 052122 RT
01761 042515 ME
01762 052116 NT
01763 005015 <15><12>
01764 040523 SA
01765 020116 N
01766 047101 AN
01767 047524 TO
01770 044516 NI
01771 026117 O,
01772 052040  T
01773 054105 EX
01774 051501 AS
01775 020040
01776 034067 78
01777 030462 21
02000 006462 2<15>
02001 005012 <12><12>
02002 044504 DI
02003 026523 S-
02004 051501 AS
02005 042523 SE
02006 041115 MB
02007 042514 LE
02010 040440  A
02011 020116 N
02012 041101 AB
02013 047523 SO
02014 052514 LU
02015 042524 TE
02016 041040  B
02017 047111 IN
02020 051101 AR
02021 020131 Y
02022 040524 TA
02023 042520 PE
02024 005015 <15><12>
02025 000000 *

       000002 .END 2
```

```
---
ADR1     001410
ADR10    001130
ADR11    001151
ADR12    001156
ADR15    001173
ADR16    001204
ADR2     001431
ADR3     001436
ADR4     001443
ADR5     001446
ADR6     001453
ADR7     000630
ADR.1    000777
ADR.2    001010
ADR.5    000616
ADR.6    000623
ADX3     000547
AREA1    001675
AREA2    001715
ARITH    001207
AT       001021
BLANK    001016
BLKS     000532
C10      001170
C12      001341
C15      001342
C4       001171
C40      000527
C60      001017
C7       001167
CMMA     001274
COMMA    001020
COUNT    000545
CR       001343
CT4      001172
DATA     000471
DISMS    001556
DMSK     000754
DOT      001022
END      000571
END2     001273
END3     000612
ERROR    000553
FAKE     001573
FIX      000757
F.2      000774
HDR      000105
HDR1     001735
IBUF     000063
INBF     001625
INBUF    000062
INDEX    001023
INSTR    000057
INTRT    001537
IO       001035
IO12     001070
IOS      001061
IOLP     001114
JUMP     000664
LONG     000635
```

```
---
LOOP    000502
MINUS   000753
MSK8    000530
OBUF    000072
OCTAL   001501
OC.2    001524
OC.5    001533
OLP.1   001510
OLP.2   001515
OTBF    001653
OTBUF   000061
P30     001406
PAGE0   000740
PC      000051
PLOOP   001375
PLUS    000752
PMSK.   001367
PRINT   001303
PRTXT   001372
PSV1    001370
PSV3    001371
PT2     000675
P.30    001323
P.35    001334
RD2     000537
RD.6    000455
READ    000440
RET     001015
RPT     000422
RSTC    000434
SHIFT   001353
SMSK    000755
SPACE   001275
START   000400
SV3     001276
SV32    001351
SVAC    001566
SVTTI   001574
SVTTO   001607
T8.1    000746
TAG8    000724
TEMP    000546
XCPU    001126
XMSK    000722
XMSK1   000723
XXX     000520
XXX2    000525
.A2     000704
.ADR2   000101
.ADR3   000102
.ADR4   000103
.ADR5   000104
.ARIT   000054
.CR     000052
.LONG   000055
.OCTA   000053
.PRIN   000056
.PRTX   000050
.SHIF   000060

---
```

```
MODE:1

                        .END

MODE:3

---

        000106          .LOC 106
   00106 020100         LDA 0,OBUF+6 ;WAIT FOR IO TO END
   00107 101004         MOV 0,0,SZR
   00110 000106         JMP .-2
   00111 063077         HALT
   00112 000002         JMP 2
        000610          .LOC 610
   00610 000106         JMP 106
        000614          .LOC 614
   00614 000106         JMP 106
        000072 OBUF=72
                        .END

---
OBUF    000072

---

MODE:
```

Correction to allow 772 to complete when END block is full.

included in object tape

PIN
LIS

USERS GROUP PROGRAM #15
PIN LIST SYSTEM
DATE SUBMITTED: JUNE, 1971
BOB WILLIAMS, W. W. WITT ASSOC.

GENERAL

THE PIN LIST SYSTEM IS DESIGNED TO ALLOW DOCUMENTATION AND IMPLEMENTATION OF LOGIC SYSTEMS WHICH USE WIRE WRAP INTERCONNECTIONS. THE NAME AND LOCATION OF EACH PIN IS ENTERED ON LINE INTO THE COMPUTER. A PAPER TAPE IS THEN MADE OF THIS SOURCE LIST. THE SOURCE LIST IS THEN READ INTO THE COMPUTER AND PAPER TAPES SORTED BY NAME AND BY LOCATION ARE MADE. THEN A FINAL LISTING OF THESE SORTED TAPES IS MADE. THE LOGIC CAN BE WIREWRAPPED UTILIZING THE LIST SORTED BY NAME AND THE LIST SORTED BY LOCATION IS USED FOR SOURCE ERROR CHECKING AND IN THE CIRCUIT DEBUG PHASE OF CHECKOUT. THE SYSTEM RUNS ON A NOVA COMPUTER WITH 4K OF CORE. TWO PROGRAMS ARE UTILIZED TO PERFORM THESE FUNCTIONS. THE PIN LIST GENERATE AND PRINT ROUTINE IS USED FOR ON LINE PIN ENTRY AND TO PRINT OUT SORTED TAPES. THE PIN LIST SORT AND PUNCH ROUTINE ACCEPTS TAPES MADE BY THE PIN LIST GENERATE AND PRINT ROUTINE AND PRODUCES PAPER TAPES SORTED BY NAME AND BY LOCATION WHICH CAN BE PRINTED BY THE PIN LIST GENERATE AND PRINT ROUTINE. THE SYSTEM CAN ACCEPT UP TO 400 INDIVIDUAL POINTS WHICH CAN BE EXPANDED BY THE ADDITION OF MORE CORE.

PIN LIST GENERATE AND PRINT

A SAMPLE LISTING AS SHOWN IN FIGURE 1 IS GENERATED BY THE PIN LIST GENERATE AND PRINT PROGRAM. WHEN THE PROGRAM IS STARTED THE USER TYPES IN THE TITLE OF THE LIST. THIS TITLE IS THEN AUTOMATICALLY PRINTED ON EACH NEW PAGE OF THE LISTING, AS WELL AS THE PAGE NUMBER AND HEADING. EACH LINE IS FORMATTED WITH A LINE NUMBER AND SPACES AUTOMATICALLY INSERTED BETWEEN THE FIELDS, SO THAT DATA CAN BE ENTERED EITHER BY KEYBOARD OR FROM PAPER TAPE. AFTER 50 LINES ARE ENTERED ON A PAGE THE PROGRAM SPACES TO THE TOP OF THE NEXT PAGE AND PRINTS THE TITLE, PAGE NUMBER AND HEADING. THE NAME FIELD IS 8 CHARACTERS BUT IT IS SUGESSTED THAT NAMES BE LIMITED TO 7 CHARACTERS SO THAT A LOGICAL NOT INDICATION SUCH AS A - MAY BE APPENDED LATER IF NECESSARY.
THE PIN LOCATION FIELD IS COMPOSED OF THREE 2 CHARACTER FIELDS.

1) BD BOARD NUMBER
2) C/R COLUMN AND ROW
3) PIN CIRCUIT PIN NUMBER

TWO INPUT EDIT MODES ARE BUILT INTO THE PROGRAM, CHARACTER DELETION AND LINE DELETION. BOTH OPERATE ONLY UPON THE CURRENT LINE DATA.
THE PREVIOUS CHARACTER ENTERED ON A LINE MAY BE DELETED BY PUNCHING THE RUB OUT KEY(RO). THE PROGRAM RESPONDS BY PRINTING THE DELETED CHARACTER.
THE ENTIRE LINE MAY BE DELETED BY TYPING / (SLASH). THE PROGRAM RESPONDS BY MOVING TO THE NEXT LINE ON THE PAGE AND REPRINTING THE LINE NUMBER AND WAITING FOR DATA ENTRY.

UPON REACHING THE END OF THE LINE THE PROGRAM WAITS FOR ONE OF THREE INPUTS.

1) CR CARRIAGE RETURN-GO TO NEXT LINE
2) RO DELETE LAST CHARACTER
3) / DELETE THE ENTIRE LINE

WHEN THE LAST INPUT HAS BEEN MADE, THE ENTIRE STORED BUFFER CAN BE PUNCHED OUT IN COMPACTED FORM (OMITTING LINE NUMBER AND FIELD SEPARATING SPACES) BY PUNCHING THE ESCAPE (ESC) KEY. THE PROGRAM RESPONDS BY SPACING TO THE TOP OF THE NEXT PAGE AND HALTING. THE USER THEN TURNS ON THE PAPER TAPE PUNCH AND PRESSES CONTINUE ON THE NOVA. A 6 INCH LEADER IS THEN PUNCHED FOLLOWED BY THE STORED DATA, FOLLOWED BY A 6 INCH TRAILER AND THE PROGRAM HALTS. THIS SOURCE TAPE MAY THEN BE USED BY THE PIN LIST SORT AND PUNCH PROGRAM.

PIN LIST SORT AND PUNCH

A SOURCE LIST TAPE PUNCHED BY THE PIN LIST GENERATE AND PRINT PROGRAM IS THE INPUT TO THE PIN LIST SORT AND PUNCH PROGRAM. THE SOURCE TAPE IS READ INTO THE COMPUTER BY THE PROGRAM BUT IS NOT PRINTED. ESC IS THEN PUNCHED AND THE PROGRAM HALTS. PUNCH IS TURNED ON AND CONTINUE IS PRESSED. A 6 INCH LEADER IS PUNCHED FOLLOWED BY THE DATA SORTED BY NAME FOLLOWED BY A 6 INCH INCH TRAILER AND THE PROGRAM HALTS. THE TAPE IS TORN OFF AND LABELED AND WITH THE PUNCH STILL ON CONTINUE IS PRESSED AGAIN. A 6 INCH LEADER FOLLOWED BY THE DATA SORTED BY LOCATION FOLLOWED BY A 6 INCH TRAILER IS PUNCHED, AND THE COMPUTER HALTS. THE TAPE IS TORN OFF AND LABELED, AND THE PUNCHED IS TURNED OFF. THE PROGRAM MUST BE RESTARTED TO SORT THE NEXT SOURCE TAPE.

PIN LIST SYSTEM
OPERATING INSTRUCTIONS

PIN LIST GENERATE AND PRINT

1) LOAD BINARY TAPE(PIN LIST GENERATE AND PRINT)AT 7777.
2) SET DATA SWITCHES TO 000002.
3) SET PAPER TO TOP OF A PAGE.
4) PRESS RESET THEN START ON THE NOVA
5) ENTER TITLE-NAME, TYPE, AND DATE. TITLE INPUT IS FREE FORM AND TERMINATED BY A CARRIAGE RETURN.
6) ENTER DATA BY KEYBOARD OR BY PAPER TAPE.
7) TERMINATE INPUT WITH AN ESC.
8) TURN ON PUNCH AND PRESS CONTINUE.
9) WHEN COMPUTER HALTS, TURN OFF PUNCH, TEAR OFF AND LABEL THE TAPE WITH THE TITLE INFO.
10) TO GENERATE THE NEXT LIST, GO TO STEP 2.

PIN LIST SORT AND PUNCH

1) LOAD BINARY TAPE(PIN LIST SORT AND PUNCH) AT 7777.
2) SET DATA SWITCHES TO 000040.
3) LOAD SOURCE TAPE IN READER AND SET READER TO START.
4) PRESS RESET THEN START ON THE NOVA.
5) AFTER TAPE HAS BEEN READ, PUNCH AN ESC.
6) TURN ON PUNCH.
7) PRESS CONTINUE TO PUNCH THE SORT BY NAME TAPE.
8) WHEN COMPUTER HALTS, TEAR OFF AND LABEL TAPE.
9) WITH PUNCH STILL ON, PRESS CONTINUE ON THE NOVA.
10) WHEN COMPUTER HALTS, TEAR OFF AND LABEL TAPE.
11) TO SORT NEXT TAPE, GO TO STEP 2.

EXAMPLE WIRELIST SOURCE LISTING 12-24-70

PAGE 1

LINE	NAME	BD	C/R	PIN		
1	FNCTB	1A	3D	Ø3		
2	START-	1A	2B	Ø3		
3	4INTA	1A	3A	Ø1		
4	AC0	1A	5B	Ø3		
5	DEVSEL	1A	6B	Ø2		
6	IOREADY	1A	2C	Ø1		
7	RGENB	1A	7C	Ø3		
8	FNCTSBBSB		1A	3D	D35D	Ø3
9	RST	1A	3B	13		
10	T20	1A	1C	15		
11	MSKO	1A	3A	10		
12	CLEAR-	1A	2B	Ø5		
13	DEVSEL	1A	4C	Ø6		
14	AC0	1A	5C	Ø1		
15	DEVCOMP	1A	7B	Ø3		
16	RST-	1A	4C	Ø4		
17	START- 1A	3C	Ø1	/		
17	START-	1A	3C	Ø1		
18	XT1	1A	1E	11		
19	AC0	/				
19	CLEAR-	1A	3A	Ø6		
20	RST	/				
20	RST	1A	4C	Ø3		
21	MSKO	1A	7C	13		
22	XT1	1A	4E	12		
23	RGENB	1A	3C	13		
24	RST-	1A	3C	11		
25	DONE-	1A	5A	Ø6		
26						

FIGURE 1

FNCTB	1A3D03
START-	1A2B03
4INTA	1A3A01
AC0	1A5B03
DEVSEL	1A6B02
IOREADY	1A2C01
RQENB	1A7C03
FNCTB	1A5D03
RST	1A3B13
T20	1A1C15
MSKO	1A3A10
CLEAR-	1A2B05
DEVSEL	1A4C06
AC0	1A5C01
DEVCOMP	1A7B03
RST-	1A4C04
START-	1A3C01
XTI	1A1E11
CLEAR-	1A3A06
RST	1A4C03
MSKO	1A7C13
XTI	1A4E12
RQENB	1A3C13
RST-	1A3C11
DONE-	1A5A06

FIGURE 2

EXAMPLE WIRELIST SORTED BY NAME 12-24-70

PAGE 1

LINE	NAME	BD	C/R	PIN
1	4INTA	1A	3A	Ø1
2	ACØ	1A	5B	Ø3
3	ACØ	1A	5C	Ø1
4	CLEAR-	1A	2B	Ø5
5	CLEAR-	1A	3A	Ø6
6	DEVCOMP	1A	7B	Ø3
7	DEVSEL	1A	4C	Ø6
8	DEVSEL	1A	6B	Ø2
9	DONE-	1A	5A	Ø6
10	FNCTB	1A	3D	Ø3
11	FNCTB	1A	5D	Ø3
12	IOREADY	1A	2C	Ø1
13	MSKO	1A	3A	10
14	MSKO	1A	7C	13
15	RQENB	1A	3C	13
16	RQENB	1A	7C	Ø3
17	RST	1A	3B	13
18	RST	1A	4C	Ø3
19	RST-	1A	3C	11
20	RST-	1A	4C	Ø4
21	START-	1A	2B	Ø3
22	START-	1A	3C	Ø1
23	T2Ø	1A	1C	15
24	XT1	1A	1E	11
25	XT1	1A	4E	12
26				

FIGURE 3

EXAMPLE WIRELIST SORTED BY PIN 12-24-70

PAGE 1

LINE	NAME	BD	C/R	PIN
1	T2Ø	1A	1C	15
2	XT1	1A	1E	11
3	START-	1A	2B	Ø3
4	CLEAR-	1A	2B	Ø5
5	IOREADY	1A	2C	Ø1
6	4INTA	1A	3A	Ø1
7	CLEAR-	1A	3A	Ø6
8	MSKO	1A	3A	1Ø
9	RST	1A	3B	13
10	START-	1A	3C	Ø1
11	RST-	1A	3C	11
12	RQENB	1A	3C	13
13	FNCTB	1A	3D	Ø3
14	RST	1A	4C	Ø3
15	RST-	1A	4C	Ø4
16	DEVSEL	1A	4C	Ø6
17	XT1	1A	4E	12
18	DONE-	1A	5A	Ø6
19	ACØ	1A	5B	Ø3
20	ACØ	1A	5C	Ø1
21	FNCTB	1A	5D	Ø3
22	DEVSEL	1A	6B	Ø2
23	DEVCOMP	1A	7B	Ø3
24	RQENB	1A	7C	Ø3
25	MSKO	1A	7C	13
26				

FIGURE 4

```
                        ;PIN LIST GENERATE AND PRINT

      000002            .LOC 2
00002 002010            JMP     @START  ;PROGRAM START ADDRESS

      000010            .LOC 10
00010 000422 START:     BEGIN           ;BEGINNING OF PROGRAM
00011 000367 GCH  :     GNZCH           ;ADDRESS OF GET NON-NULL ROUTINE

      000040            .LOC 40
00040 000000 CC   :     .-.
00041 000000 TCC  :     .-.
00042 000000 LN   :     .-.
00043 000062 LNM  :     62
00044 000000 LINE :     .-.
00045 000000 PLBC :     .-.
00046 000000 PAGE :     .-.
00047 001556 SA   :     TBL*2
00050 000000 LA   :     .-.
00051 000000 AP   :     .-.
00052 016600 APM  :     7300*2
00053 000015 CR   :     15
00054 000012 LF   :     12
00055 000033 ESC  :     33
00056 000040 SP   :     40
00057 000057 SLASH:     57
00060 000177 RO   :     177
00061 000010 C10  :     10
00062 000016 C16  :     16
      000060 C177=RO
00063 000377 C377 :     377
00064 000556 LTBLA:     LMSD
00065 000004 LWC  :     4
00066 000552 PTBLA:     PMSD
00067 000546 PGA  :     CP
00070 000010 PWC  :     10
00071 001436 TTBLA:     TTBL*2
00072 001344 HTBLA:     HTBL*2
00073 000070 HWC  :     70
00074 000012 LFC  :     12
00075 177700 LCNT :     -100
00076 000000 R1   :     .-.
00077 000000 R2   :     .-.
00100 000000 R3   :     .-.
00101 000000 R4   :     .-.
00102 000000 R5   :     .-.
00103 000000 R6   :     .-.
00104 000000 R7   :     .-.
00105 000000 R8   :     .-.
00106 000000 R9   :     .-.
00107 000000 R10  :     .-.
00110 000000 R11  :     .-.
00111 000000 R12  :     .-.
```

```
00112 054145 BIDAS:  STA     3,SAVE3 ;SAVE RETURN
00113 050146         STA     2,TAD   ;SAVE TABLE ADDRESS
00114 050147         STA     2,TBLAD ;SAVE TABLE ADDRESS AGAIN
00115 034153         LDA     3,ATTBL ;GET TENS TABLE ADDRESS
00116 020152 DLOOP:  LDA     0,ZERO  ;GET ASCII ZERO
00117 031400         LDA     2,0,3   ;GET TENS
00120 146422 ILOOP:  SUBZ    2,1,SZC ;DOES POWER OF TEN GO IN?
00121 101401         INC     0,0,SKP ;YES-BUMP DIGIT
00122 147001         ADD     2,1,SKP ;NO-RESTORE REMAINDER
00123 000120         JMP     ILOOP   ;CONTINUE SUBTRACTING
00124 042147         STA     0,@TBLAD;STORE DIGIT
00125 010147         ISZ     TBLAD   ;INCREMENT TABLE ADDRESS
00126 175400         INC     3,3     ;BUMP TENS TABLE ADDRESS
00127 151234         MOVZR#  2,2,SZR ;TEST TO SEE IF 1 WAS DONE
00130 000116         JMP     DLOOP   ;NO-CONTINUE
00131 034146         LDA     3,TAD   ;GET TABLE ADDRESS
00132 020152         LDA     0,ZERO  ;GET ASCII ZERO
00133 024154         LDA     1,COUNT ;GET -3
00134 031400 DLZLP:  LDA     2,0,3   ;GET DIGIT
00135 142414         SUB#    2,0,SZR ;IS IT A LEADING ZERO?
00136 002145         JMP     @SAVE3  ;NO-RETURN
00137 030151         LDA     2,SPACE ;YES-GET ASCII SPACE
00140 051400         STA     2,0,3   ;REPLACE ZERO WITH SPACE
00141 175400         INC     3,3     ;BUMP ADDRESS
00142 125404         INC     1,1,SZR ;IS COUNT DONE?
00143 000134         JMP     DLZLP   ;NO-TEST NEXT DIGIT
00144 002145         JMP     @SAVE3  ;DONE-RETURN

00145 000000 SAVE3:  .-.
00146 000000 TAD  :  .-.
00147 000000 TBLAD:  .-.
00150 000055 MINUS:  55
00151 000040 SPACE:  40
00152 000060 ZERO :  60
00153 000155 ATTBL:  TTBLE
00154 177775 COUNT:  -3

      000012         .RDX 10
00155 001750 TTBLE:   1000   ;10**3
00156 000144            100  ;10**2
00157 000012             10  ;10**1
00160 000001              1  ;10**0

      000010         .RDX 8
```

```
;;;
00161 054076 GETL :  STA     3,R1      ;SAVE RETURN
00162 102400         SUB     0,0       ;
00163 040041         STA     0,TCC     ;SET TITLE CHAR COUNT=0
00164 004260         JSR     PCRLF     ;PRINT CR&LF
00165 006011 TLOOP:  JSR     0GCH      ;GET CHARACTER
00166 024053         LDA     1,CR      ;TEST FOR
00167 122415         SUB#    1,0,SNR   ;        CARRAIGE RETURN
00170 000200         JMP     TDONE     ;YES-TITLE INPUT DONE
00171 004266         JSR     PRT1      ;NO-PRINT CHARACTER
00172 030071         LDA     2,TTBLA   ;GET TITLE ADDRESS
00173 024041         LDA     1,TCC     ;GET CHARACTER COUNT
00174 133000         ADD     1,2       ;ADD TO GET ADDRESS
00175 004231         JSR     PBYTE     ;STORE CHARACTER
00176 010041         ISZ     TCC       ;TCC=TCC+1
00177 000165         JMP     TLOOP     ;GET NEXT CHARACTER
00200 004260 TDONE:  JSR     PCRLF     ;PRINT CR&LF
00201 004260         JSR     PCRLF     ; AND AGAIN
00202 002076         JMP     0R1       ;DONE-RETURN

00203 054077 PHDG :  STA     3,R2      ;SAVE RETURN
00204 024046         LDA     1,PAGE    ;GET PAGE COUNT
00205 030066         LDA     2,PTBLA   ;GET PAGE TABLE ADDRESS
00206 004112         JSR     BIDAS     ;CONVERT BINARY TO ASCII
00207 030067         LDA     2,PGA     ;GET PAGE P ADDRESS
00210 024070         LDA     1,PWC     ;GET WORD COUNT
00211 004310         JSR     PTTU      ;PRINT PAGE &NUMBER
00212 004260         JSR     PCRLF     ;PRINT 2
00213 004260         JSR     PCRLF     ;     CR&LF'S
00214 030072         LDA     2,HTBLA   ;GET HEADING ADDRESS
00215 024073         LDA     1,HWC     ;GET HEADING BYTE COUNT
00216 004320         JSR     PTTP      ;PRINT HEADING
00217 002077         JMP     0R2       ;DONE-RETURN

00220 054100 PLIN :  STA     3,R3      ;SAVE RETURN
00221 024044         LDA     1,LINE    ;GET LINE NUMBER
00222 030064         LDA     2,LTBLA   ;GET LINE TABLE ADDRESS
00223 004112         JSR     BIDAS     ;CONVERT BINARY TO ASCII
00224 024065         LDA     1,LWC     ;GET LINE WORD COUNT
00225 030064         LDA     2,LTBLA   ;
00226 004310         JSR     PTTU      ;PRINT LINE NUMBER
00227 004253         JSR     P2SP      ;PRINT 2 SPACES
00230 002100         JMP     0R3       ;DONE-RETURN

00231 054101 PBYTE:  STA     3,R4      ;SAVE RETURN
00232 151220         MOVER   2,2       ;ORIENT WORD ADDRESS-BP IN CARRY
00233 025000         LDA     1,0,2     ;GET WORD
00234 034063         LDA     3,C377    ;GET BYTE MASK
00235 101002         MOV     0,0,SZC   ;ASSUME LEFT
00236 175301         MOVS    3,3,SKP   ;NO SWAP MASK
00237 101300         MOVS    0,0       ;YES-SWAP BYTE
00240 167400         AND     3,1       ;MASK OLD BYTE
00241 107000         ADD     0,1       ;ADD IN NEW BYTE
00242 045000         STA     1,0,2
00243 002101         JMP     0R4       ;DONE-RETURN

00244 151220 GBYTE:  MOVER   2,2       ;ORIENT WORD ADDRESS-BP IN CARRY
00245 021000         LDA     0,0,2     ;GET WORD
00246 101302         MOVS    0,0,SZC   ;ASSUME LEFT
00247 101300         MOVS    0,0       ;NO-RIGHT
```

```
00250 030063          LDA     2,C377  ;GET BYTE MASK
00251 143400          AND     2,0     ;BITS 8-15 REMAIN
00252 001400          JMP     0,3     ;DONE-RETURN

00253 054102 P2SP :   STA     3,R5    ;SAVE RETURN
00254 020056          LDA     0,SP    ;GET SPACE
00255 004266          JSR     PRT1    ;PRINT 2
00256 004266          JSR     PRT1    ;      SPACES
00257 002102          JMP     @R5     ;DONE-RETURN

00260 054103 PCRLF:   STA     3,R6    ;SAVE RETURN
00261 020053          LDA     0,CR    ;GET CARRIAGE RETURN
00262 004266          JSR     PRT1    ;PRINT IT
00263 020054          LDA     0,LF    ;GET LINE FEED
00264 004266          JSR     PRT1    ;PRINT IT
00265 002103          JMP     @R6     ;DONE-RETURN

00266 063611 PRT1 :   SKPDN   TTO     ;SKIP IF TTO DONE
00267 000266          JMP     .-1     ;NOT DONE
00270 061111          DOAS    0,TTO   ;PRINT
00271 001400          JMP     0,3     ;DONE-RETURN

00272 054104 MLF  :   STA     3,R7    ;SAVE RETURN
00273 020054          LDA     0,LF    ;GET LINE FEED
00274 124400          NEG     1,1     ;NEGATE WORD COUNT
00275 004266 MLP  :   JSR     PRT1    ;PRINT LINE FEED
00276 125404          INC     1,1,SZR ;BUMP WORD COUNT
00277 000275          JMP     MLP     ;NOT DONE
00300 002104          JMP     @R7     ;DONE-RETURN

00301 054105 LDR  :   STA     3,R8    ;SAVE RETURN
00302 024075          LDA     1,LCNT  ;GET LEADER COUNT
00303 102400          SUB     0,0     ;GET NULL
00304 004266 LLP  :   JSR     PRT1    ;PRINT NULL
00305 125404          INC     1,1,SZR ;BUMP WORD COUNT
00306 000304          JMP     LLP     ;NOT DONE
00307 002105          JMP     @R8     ;DONE-RETURN

00310 054106 PTTU :   STA     3,R9    ;SAVE RETURN
00311 124400          NEG     1,1     ;NEGATE WORD COUNT
00312 021000 PTLP :   LDA     0,0,2   ;GET WORD
00313 004266          JSR     PRT1    ;PRINT IT
00314 125405          INC     1,1,SNR ;BUMP WORD COUNT
00315 002106          JMP     @R9     ;DONE-RETURN
00316 151400          INC     2,2     ;BUMP ADDRESS
00317 000312          JMP     PTLP    ;GET NEXT WORD

00320 054107 PTTP :   STA     3,R10   ;SAVE RETURN
00321 124400          NEG     1,1     ;NEGATE BYTE COUNT
00322 151222          MOVZR   2,2,SZC ;ORIENT ADDRESS-TEST BP
00323 000331          JMP     PRTRT   ;PRINT RIGHT FIRST
00324 021000 PLP  :   LDA     0,0,2   ;GET WORD
00325 101300          MOVS    0,0     ;SWAP TO PRINT LEFT
00326 004266          JSR     PRT1    ;PRINT LEFT
00327 125405          INC     1,1,SNR ;BUMP BYTE COUNT
00330 002107          JMP     @R10    ;DONE-RETURN
00331 021000 PRTRT:   LDA     0,0,2   ;GET WORD
00332 004266          JSR     PRT1    ;PRINT RIGHT
00333 125405          INC     1,1,SNR ;BUMP CHAR COUNT
00334 002107          JMP     @R10    ;DONE-RETURN
```

```
---
00335 151400          INC     2,2     ;BUMP WORD ADDRESS
00336 000324          JMP     PLP     ;PRINT NEXT

00337 054110 PTTL :   STA     3,R11   ;SAVE RETURN
00340 024041          LDA     1,TCC   ;GET TITLE BYTE COUNT
00341 030071          LDA     2,TTBLA ;GET TITLE ADDRESS
00342 004320          JSR     PTTP    ;PRINT TITLE
00343 004260          JSR     PCRLF   ;PRINT 2
00344 004260          JSR     PCRLF   ;      CRLF'S
00345 002110          JMP     @R11    ;DONE-RETURN
```

```
???
00346 024040 DELCH: LDA     1,CC     ;GET LINE CHAR COUNT
00347 125005        MOV     1,1,SNR  ;TEST FOR ZERO
00350 000467        JMP     GET      ;ZERO-RETURN
00351 014040        DSZ     CC       ;DECREMENT CC
00352 000353        JMP     .+1      ;
00353 014051        DSZ     AP       ;DECREMENT ADDRESS POINTER
00354 030051        LDA     2,AP     ;GET POINTER
00355 004244        JSR     GBYTE    ;GET BYTE
00356 004266        JSR     PRT1     ;PRINT IT
00357 000460        JMP     GET      ;DONE-RETURN

00360 024040 DELN : LDA     1,CC     ;GET LINE CHAR COUNT
00361 030051        LDA     2,AP     ;GET ADDRESS POINTER
00362 132400        SUB     1,2      ;SUBTRACT LINE COUNT
00363 050051        STA     2,AP     ;STORE NEW ADDRESS POINTER
00364 014044        DSZ     LINE     ;DECREMENT LINE NUMBER
00365 000366        JMP     .+1      ;
00366 000524        JMP     B        ;TEST LN,ETC

00367 063610 GNZCH: SKPDN   TTI      ;SKIP IF INPUT DONE
00370 000367        JMP     .-1      ;NOT DONE
00371 060510        DIAS    0,TTI    ;DONE-READ CHARACTER
00372 024060        LDA     1,C177   ;GET MASK
00373 123405        AND     1,0,SNR  ;BITS 9-15 LEFT-TEST FOR NULL
00374 000367        JMP     GNZCH    ;NULL-TRY AGAIN
00375 001400        JMP     0,3      ;OK-RETURN

00376 054111 PRINT: STA     3,R12    ;SAVE RETURN
00377 004301        JSR     LDR      ;PUNCH LEADER
00400 014044        DSZ     LINE     ;DECREMENT LINE
00401 101001        MOV     0,0,SKP  ;OK-LINE NOT ZERO
00402 004542        JSR     ENDPR    ;LINE=0-GO TO END HALT
00403 020047        LDA     0,SA     ;GET START ADDRESS
00404 040050        STA     0,LA     ;SAVE FOR PRINT ROUTINE
00405 020062 PLOOP: LDA     0,C16    ;USE C16
00406 040045        STA     0,PLBC   ;    FOR LINE BYTE COUNT
00407 030050 LLOOP: LDA     2,LA     ;GET LINE ADDRESS
00410 004244        JSR     GBYTE    ;GET BYTE
00411 004266        JSR     PRT1     ;PRINT IT
00412 010050        ISZ     LA       ;LA=LA+1
00413 014045        DSZ     PLBC     ;DECREMENT LINE BYTE COUNT
00414 000773        JMP     LLOOP    ;NOT END OF LINE
00415 004260        JSR     PCRLF    ;END OF LINE
00416 014044        DSZ     LINE     ;DECREMENT LINE COUNT
00417 000766        JMP     PLOOP    ;NOT DONE
00420 004301        JSR     LDR      ;DONE-PUNCH TRAILER
00421 002111        JMP     @R12     ;DONE-RETURN
```

```
---
↑↑↑
00422 102400 BEGIN:  SUB     0,0      ;CLEAR AC0
00423 061111         DOAS    0,TTO    ;START TTO
00424 060110         NIOS    TTI      ;START TTI
                                      ;INITIALIZE-
00425 040040         STA     0,CC     ;   LINE BYTE COUNT
00426 040042         STA     0,LN     ;  PAGE LINE COUNT
00427 101400         INC     0,0      ;AC(0)=1
00430 040044         STA     0,LINE   ;SET LINE =1
00431 040046         STA     0,PAGE   ;AND PAGE=1
00432 020047         LDA     0,SA     ;SET
00433 040051         STA     0,AP     ;   ADDRESS POINTER=START ADDRES
00434 004161         JSR     GETL     ;GET TITLE-(KEYBOARD ENTRY
00435 004203         JSR     PHDG     ;PRINT PAGE AND HEADING
00436 004220         JSR     PLIN     ;PRINT LINE NUMBER
00437 004367 GET  :  JSR     GNZCH    ;GET CHARACTER
00440 024055         LDA     1,ESC    ;TEST FOR
00441 122415         SUB#    1,0,SNR  ;   END OF INPUT
00442 000471         JMP     IDONE    ;YES-INPUT DONE
00443 024056         LDA     1,SP     ;NO-TEST,CHARACTER MUST
00444 122433         SUBZ#   1,0,SNC  ;   BE>=SPACE(40)
00445 000772         JMP     GET      ;NO-TRY AGAIN
00446 024060         LDA     1,RO     ;OK-TEST FOR RUBOUT
00447 122415         SUB#    1,0,SNR  ;DELETE CHARACTER?
00450 000346         JMP     DELCH    ;YES
00451 004266         JSR     PRT1     ;NO-PRINT IT
00452 024057         LDA     1,SLASH  ;TEST FOR
00453 122415         SUB#    1,0,SNR  ;DELETE LINE?
00454 000360         JMP     DELN     ;YES
00455 030051         LDA     2,AP     ;NO-GET ADDRESS POINTER
00456 004231         JSR     PBYTE    ;STORE BYTE
00457 010051         ISZ     AP       ;AP=AP+1
00460 020051         LDA     0,AP     ;GET AP
00461 024052         LDA     1,APM    ;SEE IF END OF TABLE
00462 106433         SUBZ#   0,1,SNC  ;   HAS BEEN REACHED
00463 000450         JMP     IDONE    ;YES-TERMINATE INPUT
00464 010040         ISZ     CC       ;BUMP LINE CHAR. COUNT
00465 020040         LDA     0,CC     ;GET CC
00466 024061         LDA     1,C10    ;SEE IF END OF NAME
00467 106032         ADCZ#   0,1,SZC  ;    HAS BEEN REACHED
00470 000747         JMP     GET      ;NO-GET NEXT
00471 101212         MOVR#   0,0,SZC  ;YES-IS CC EVEN?
00472 000745         JMP     GET      ;NO
00473 024062         LDA     1,C16    ;YES-HAS LINE END
00474 122415         SUB#    1,0,SNR  ;   BEEN REACHED?
00475 000403         JMP     NLINE    ;YES-NEW LINE
00476 004253         JSR     P2SP     ;NO-PRINT 2 SPACES
00477 000740         JMP     GET      ;GET NEXT CHARACTER

00500 004367 NLINE:  JSR     GNZCH    ;GET CHARACTER
00501 024060         LDA     1,RO     ;MUST BE A
00502 122415         SUB#    1,0,SNR  ;    RUBOUT
00503 000346         JMP     DELCH    ;YES-DELETE CHARACTER
00504 024057         LDA     1,SLASH  ;   OR A
00505 122415         SUB#    1,0,SNR  ;    SLASH
00506 000360         JMP     DELN     ;YES-DELETE LINE
00507 024053         LDA     1,CR     ;   OR A
00510 122414         SUB#    1,0,SZR  ;    CARRIAGE RETURN
00511 000767         JMP     NLINE    ;ITS NONE OF THE ABOVE
00512 004260 B    :  JSR     PCRLF    ;ITS A CR
```

```
00513 010042           ISZ     LN        ;LN=LN+1
00514 020042           LDA     0,LN      ;GET LN
00515 024043           LDA     1,LNM     ;AND MAX LINES PER PAGE
00516 106404           SUB     0,1,SZR   ;IS LN=LNM?
00517 000407           JMP     A         ;NO
00520 044042           STA     1,LN      ;YES-SET LN=0
00521 024074           LDA     1,LFC     ;GET LINE FEED COUNT
00522 004272           JSR     MLF       ;SPACE TO TOP OF PAGE
00523 010046           ISZ     PAGE      ;BUMP PAGE NUMBER
00524 004337           JSR     PTTL      ;PRINT TITLE
00525 004203           JSR     PHDG      ;PRINT HEADING
00526 010044 A     :   ISZ     LINE      ;LINE=LINE+1
00527 004220           JSR     PLIN      ;PRINT LINE NUMBER
00530 102400           SUB     0,0       ;
00531 040040           STA     0,CC      ;SET CC=0
00532 000705           JMP     GET       ;GET NEXT CHARACTER

00533 024042 IDONE:    LDA     1,LN      ;GET LINE NUMBER
00534 020043           LDA     0,LNM     ;GET MAX LINE NUMBER
00535 122400           SUB     1,0       ;GET LINES TO MAX
00536 024074           LDA     1,LFC     ;ADD
00537 107000           ADD     0,1       ;   LINE FEED COUNT
00540 004272           JSR     MLF       ;SPACE TO TOP OF PAGE
00541 004260           JSR     PCRLF     ;PRINT CR&LF
00542 063077           HALT              ;TURN ON PUNCH
00543 004376           JSR     PRINT     ;PRINT STORED LIST
00544 063077 ENDPR:    HALT              ;END OF PROGRAM
00545 000777           JMP     .-1       ; CAN'T CONTINUE
```

```
00546 000120 CP   :  120      ;P
00547 000101 CA   :  101      ;A
00550 000107 CG   :  107      ;G
00551 000105 CE   :  105      ;E
00552 000000 PMSD :  .-.      ;10**3-MOST SIGN.DIG.PAGE NUMBER
00553 000000         .-.      ;10**2
00554 000000         .-.      ;10**1
00555 000000         .-.      ;10**0

00556 000000 LMSD :  .-.      ;10**3-MOST SIGN.DIG.LINE NUMBER
00557 000000         .-.      ;10**2
00560 000000         .-.      ;10**1
00561 000000         .-.      ;10**0

      000001         .TXTM 1  ;STORE TEXT LEFT/RIGHT
00562 046111 HTBL :  .TXT"LI
00563 047105 NE
00564 020040
00565 020040
00566 047101 NA
00567 046505 ME
00570 020040
00571 020040
00572 041104 BD
00573 020103  C
00574 027522 /R
00575 020120  P
00576 044516 IN
00577 006412 <15><12>
00600 026455 --
00601 026455 --
00602 026455 --
00603 026455 --
00604 026455 --
00605 026455 --
00606 026455 --
00607 026455 --
00610 026455 --
00611 026455 --
00612 026455 --
00613 026455 --
00614 026455 --
00615 006412 <15><12>
00616 000000 "


      000050 TTBL :  .BLK 50 ;TITLE TEXT STORAGE

00667 000000 TBL  :  .-.      ;INFORMATION STORAGE TABLE

                     .END
```

```
---
A       000526
AP      000051
APM     000052
ATTBL   000153
B       000512
BEGIN   000422
BIDAS   000112
C10     000061
C16     000062
C177    000060
C377    000063
CA      000547
CC      000040
CE      000551
CG      000550
COUNT   000154
CP      000546
CR      000053
DELCH   000346
DELN    000360
DLOOP   000116
DLYLP   000134
ENDPR   000544
ESC     000055
GBYTE   000244
GCH     000011
GET     000437
GETL    000161
GNXCH   000367
HTBL    000562
HTBLA   000072
HWC     000073
IDONE   000533
ILOOP   000120
LA      000050
LCNT    000075
LDR     000301
LF      000054
LFC     000074
LINE    000044
LLOOP   000407
LLP     000304
LMSD    000556
LN      000042
LNM     000043
LTBLA   000064
LWC     000065
MINUS   000150
MLF     000272
MLP     000275
NLINE   000500
P2SP    000253
PAGE    000046
PBYTE   000231
PCRLF   000260
PGA     000067
PHDG    000203
PLBC    000045
PLIN    000220
PLOOP   000405
```

```
PLP     000324
PMSD    000552
PRINT   000376
PRT1    000266
PRTRT   000331
PTBLA   000066
PTLP    000312
PTTL    000337
PTTP    000320
PTTU    000310
PWC     000070
R1      000076
R10     000107
R11     000110
R12     000111
R2      000077
R3      000100
R4      000101
R5      000102
R6      000103
R7      000104
R8      000105
R9      000106
RO      000060
SA      000047
SAVE3   000145
SLASH   000057
SP      000056
SPACE   000151
START   000010
TAD     000146
TBL     000667
TBLAD   000147
TCC     000041
TDONE   000200
TLOOP   000165
TTBL    000617
TTBLA   000071
TTBLE   000155
ZERO    000152
```

```
        000040          .LOC 40
00040 102400 BEGIN:  SUB     0,0     ;
00041 040131         STA     0,IBC   ;SET IBC=0
00042 040145         STA     0,SF    ;    SF =0
00043 020144         LDA     0,SA    ; AND
00044 040132         STA     0,AP    ;    AP =SA
00045 004150 GET  :  JSR     GNECH   ;GET NON NULL CAHRACTER
00046 024134         LDA     1,ESC   ;TEST
00047 122415         SUB#    1,0,SNR ;    FOR ESCAPE
00050 000101         JMP     IDONE   ;YES-INPUT DONE
00051 024135         LDA     1,CR    ;NO-TEST
00052 122415         SUB#    1,0,SNR ;       FOR LINE DONE
00053 000065         JMP     ENDLN   ;YES
00054 030132         LDA     2,AP    ;NO
00055 004162         JSR     PBYTE   ;STORE BYTE
00056 010131         ISZ     IBC     ;BUMP INPUT LINE BYTE COUNT
00057 010132         ISZ     AP      ;BUMP ADDRESS POINTER
00060 030132         LDA     2,AP    ;CHECK TO SEE
00061 024133         LDA     1,APM   ;   IF END OF TABLE
00062 146433         SUBZ#   2,1,SNC ;       HAS BEEN REACHED
00063 000101         JMP     IDONE   ;YES-SAME AS INPUT DONE
00064 000045         JMP     GET     ;NO - CONTINUE

00065 004150 ENDLN:  JSR     GNECH   ;GET CHARACTER
00066 024136         LDA     1,LF    ;TEST
00067 122414         SUB#    1,0,SZR ;    FOR LINE FEED
00070 004127         JSR     ERR     ;NO-ERROR
00071 024131         LDA     1,IBC   ;TEST FOR
00072 030146         LDA     2,LBC   ;   PROPER LINE
00073 146414         SUB#    2,1,SZR ;       BYTE COUNT
00074 004127         JSR     ERR     ;NO-ERROR
00075 102400         SUB     0,0     ;OK-
00076 040131         STA     0,IBC   ;SET IBC = 0
00077 010145         ISZ     SF      ;BUMP SORT FIELD COUNT
00100 000045         JMP     GET     ;GET NEXT LINE

00101 020146 IDONE:  LDA     0,LBC   ;SET
00102 040147         STA     0,SFBC  ;   SFBC=LBC
00103 020143         LDA     0,TBL   ;GET TABLE ADDRESS
00104 063077         HALT             ;TURN ON PUNCH
00105 006140         JSR     @ASORT  ;SORT
00106 004127         JSR     ERR     ;SORT ERROR
00107 040141         STA     0,SBA   ;SAVE SORTED BLOCK ADDRESS
00110 126400         SUB     1,1     ;ADDRESS MODIFIER=0
00111 004177         JSR     PRINT   ;PRINT BY NAME
00112 020144         LDA     0,SA    ;DEVELOP START ADDRESS
00113 024142         LDA     1,BN    ;   FOR SORT
00114 123000         ADD     1,0     ;      BY PIN
00115 040144         STA     0,SA    ;       AND STORE IT
00116 020137         LDA     0,PBC   ;GET PIN BYTE COUNT
00117 040147         STA     0,SFBC  ;   AND STORE IT
00120 020143         LDA     0,TBL   ;GET TABLE ADDRESS
00121 063077         HALT             ;
00122 006140         JSR     @ASORT  ;SORT
00123 004127         JSR     ERR     ;SORT ERROR
00124 040141         STA     0,SBA   ;SAVE SORTED BLOCK ADDRESS
00125 024142         LDA     1,BN    ;GET OFFSET BYTE COUNT
```

```
00131 000000 IBC   :   .-.
00132 000000 AP    :   .-.
00133 014000 APM   :   6000*2
00134 000033 ESC   :   33
00135 000015 CR    :   15
00136 000012 LF    :   12
00137 000006 PBC   :   6
00140 006000 ASORT:    6000
00141 000000 SBA   :   .-.
00142 000010 BN    :   10
00143 000144 TBL   :   SA
00144 000554 SA    :   TBL A*2
00145 000000 SF    :   .-.
00146 000016 LBC   :   16
00147 000000 SFBC  :   .-.
```

```
00151 063610           SKPDN   TTI     ;SKIP IF TTI DONE
00152 000151           JMP     .-1     ;NOT DONE
00153 060610           DIAC    0,TTI   ;DONE-GET CHARACTER
00154 101005           MOV     0,0,SNR ;TEST FOR NULL
00155 000150           JMP     CNZCH   ;NULL-TRY AGAIN
00156 024161           LDA     1,C177  ;GET MASK
00157 123400           AND     1,0     ;ELIMINATE PARITY
00160 001400           JMP     0,3     ;DONE
00161 000177 C177 :    177

00162 054176 PBYTE:    STA     3,SAVE3 ;SAVE RETURN
00163 151220           MOVZR   2,2     ;ORIENT WORD ADDRESS-BP IN CARRY
00164 025000           LDA     1,0,2   ;GET WORD
00165 034175           LDA     3,C377  ;GET BYTE MASK
00166 101002           MOV     0,0,SZC ;ASSUME LEFT
00167 175301           MOVS    3,3,SKP ;NO-SWAP MASK
00170 101300           MOVS    0,0     ;YES-SWAP BYTE
00171 167400           AND     3,1     ;MASK OLD BYTE
00172 107000           ADD     0,1     ;ADD IN NEW BYTE
00173 045000           STA     1,0,2   ;STORE WORD
00174 002176           JMP     @SAVE3  ;RETURN
00175 000377 C377 :    377
00176 000000 SAVE3:    .-.

00177 054265 PRINT:    STA     3,SAV3  ;SAVE RETURN
00200 044260           STA     1,PBN   ;SAVE LINE OFFSET
00201 004237           JSR     LDR     ;PRINT LEADER
00202 020145           LDA     0,SF    ;INITIALIZE
00203 040262           STA     0,PSF   ;       NUMBER OF SORT FIELDS
00204 020141           LDA     0,SBA   ;       AND
00205 040263           STA     0,PSBA  ;       SORTED BLOCK ADDRESS
00206 020146 PLOOP:    LDA     0,LBC   ;       AND
00207 040261           STA     0,PLBC  ;LINE BYTE COUNT
00210 032263           LDA     2,@PSBA ;GET ADDRESS
00211 024260           LDA     1,PBN   ;GET OFFSET
00212 132400           SUB     1,2     ;GET TO LINE BEGINING
00213 050264           STA     2,PBA   ;SAVE 1ST BYTE ADDRESS
00214 030264 LLOOP:    LDA     2,PBA   ;GET BYTE ADDRESS
00215 004250           JSR     GBYTE   ;GET BYTE
00216 004233           JSR     PRT1    ;PRINT IT
00217 010264           ISZ     PBA     ;BUMP BYTE ADDRESS
00220 014261           DSZ     PLBC    ;DECREMENT LINE BYTE COUNT
00221 000214           JMP     LLOOP   ;NOT DONE
00222 020135           LDA     0,CR    ;END OF LINE
00223 004233           JSR     PRT1    ;PRINT CARRIAGE RETURN
00224 020136           LDA     0,LF    ;
00225 004233           JSR     PRT1    ;PRINT LINE FEED
00226 010263           ISZ     PSBA    ;BUMP FIELD ADDRESS
00227 014262           DSZ     PSF     ;DECREMENT SORT FIELD COUNT
00230 000206           JMP     PLOOP   ;NOT DONE
00231 004237           JSR     LDR     ;PRINT TRAILER
00232 002265           JMP     @SAV3   ;DONE

00233 061111 PRT1 :    DOAS    0,TTO   ;PRINT BYTE
00234 063611           SKPDN   TTO     ;SKIP IF TTO DONE
00235 000234           JMP     .-1     ;NOT DONE
00236 001400           JMP     0,3     ;DONE
```

```
00240 024246           LDA     1,COUNT ;GET LEADER COUNT
00241 102400           SUB     0,0     ;CLEAR AC0 FOR NULL
00242 004233 LLP  :    JSR     PRT1    ;PRINT NULL
00243 125404           INC     1,1,SZR ;BUMP NEG LEADRE COUNT
00244 000242           JMP     LLP     ;NOT DONE
00245 002247           JMP     @S3     ;DONE-RETURN
00246 177700 COUNT:    -100
00247 000000 S3   :    .-.

00250 151220 GBYTE:    MOVZR   2,2     ;ORIENT WORD ADDRESS BP IN CARRY
00251 021000           LDA     0,0,2   ;GET WORD
00252 101302           MOVS    0,0,SZC ;ASSUME LEFT BYTE
00253 101300           MOVS    0,0     ;NO-RIGHT
00254 030257           LDA     2,X377  ;GET MASK
00255 143400           AND     2,0     ;BITS 8-15 LEFT
00256 001400           JMP     0,3     ;RETURN
00257 000377 X377 :    377

00260 000000 PBN  :    .-.
00261 000000 PLBC :    .-.
00262 000000 PSF  :    .-.
00263 000000 PSBA :    .-.
00264 000000 PBA  :    .-.
00265 000000 SAV3 :    .-.

00266 000000 TBLA :    .-.

                       .END
```

```
A          000132
ARM        000133
ASORT      000140
BEGIN      000040
BN         000142
C177       000161
C377       000175
COUNT      000246
CR         000135
ENDLN      000065
ERR        000127
ESC        000134
GBYTE      000250
GET        000045
GNZCH      000150
IBC        000131
IDONE      000101
LBC        000146
LDR        000237
LF         000136
LLOOP      000214
LLP        000242
PBA        000264
PBC        000137
PBN        000260
PBYTE      000162
PLBC       000261
PLOOP      000206
PRINT      000177
PRT1       000233
PSBA       000263
PSF        000262
S3         000247
SA         000144
SAV3       000265
SAVE3      000176
SBA        000141
SF         000145
SFBC       000147
TBL        000143
TBLA       000266
X377       000257
```

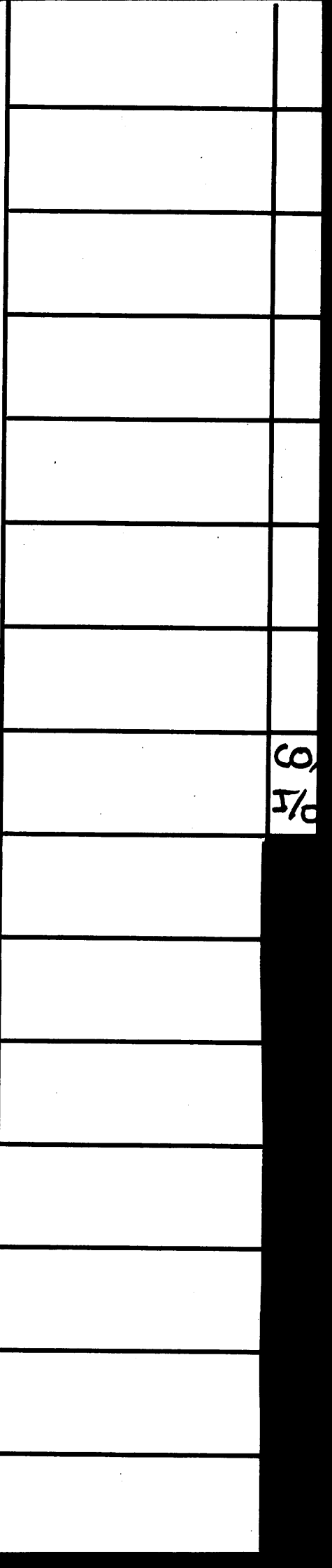
```
;SORT
;SORTS FIELDS OF PACKED 8 BIT
;BYTES.  THE FIELDS MAY BE WITHIN
;A MAJOR FIELD AND BOTH THE SORT FIELD
;AND THE MAJOR FIELD MAY BE OF EVEN
;OR ODD NUMBER OF BYTES. THE ROUTINE
;RETURNS THE ADDRESS OF THE SORT TABLE
;WHICH CONTAINS THE 1ST BYTE ADDRESS
;OF THE SORT FIELDS IN SORTED ORDER.
;CALLING SEQUENCE-
;   AC0-ADDRESS OF THE ARGUMENT TABLE-T
;     T  -ADDRESS OF 1ST BYTE, 1ST SORT FIELD
;     T+1-NUMBER OF FIELDS TO SORT
;     T+2-NUMBER OF BYTES IN MAJOR FIELD
;     T+3-NUMBER OF BYTES IN SORT FIELD
;
;RETURN
;     CALL-ERROR RETURN
;     CALL+1-NORMAL RETURN
;      AC0-ADDRESS OF SORT TABLE
;
;ALL AC'S ARE LOST
;
;STORAGE REQUIREMENTS
;   PROGRAM-(232)8 PLUS
;     SORT TABLE(NUMBER OF FIELDS TO SORT+
;              NUMBER OF BYTES IN SORT FIELD)8
;ERROR RETURN IF (MAX) IS EXCEEDED
;MAX SET AT (7400)8
;SORT SUBROUTINES INCLUDE
;   ADGEN, LOW, AND GBYTE.


.EOT
```

```
06000 054513 SORT :  STA     3, SAVE3 ;SAVE RETURN
06001 115000         MOV     0, 3     ;MOVE TABLE ADDRESS TO INDEX
06002 021400         LDA     0, 0, 3  ;GET 1ST BYTE ADDRESS
06003 040511         STA     0, SA    ;STORE AT START ADDRESS
06004 021401         LDA     0, 1, 3  ;GET FIELD COUNT
06005 040510         STA     0, FC    ;STORE IT
06006 031402         LDA     2, 2, 3  ;GET MAJOR FIELD BYTE COUNT
06007 050507         STA     2, MFBC  ;STORE IT
06010 025403         LDA     1, 3, 3  ;GET SORT FIELD BYTE COUNT
06011 132433         SUBZ#   1, 2, SNC ;IS SFBC<=MFBC?
06012 002501         JMP     @SAVE3   ;NO-ERROR
06013 125005         MOV     1, 1, SNR ;OK-IS SFBC=0?
06014 002477         JMP     @SAVE3   ;YES-ERROR
06015 044502         STA     1, SFBC  ;NO-STORE SFBC
06016 014501         DSZ     SFBC     ;THEN DECREMENT IT
06017 000401         JMP     .+1      ;IF SFBC WAS 1,NOW 0
06020 030500         LDA     2, XAPTB ;GET ADDRESS OF POINTER TABLE
06021 133000         ADD     1, 2     ;ADD TO GET PERMANENT
06022 050477         STA     2, SEAP  ;    SORT ADDRESS TABLE 1ST ADDRS
06023 050477         STA     2, SBA   ;INITIALIZE WORKING SBA
06024 143000         ADD     2, 0     ;TO GET LAST TABLE ADDRESS
06025 024476         LDA     1, MAX   ;GET MAX ADDRESS
06026 106433         SUBZ#   0, 1, SNC ;MAKE SURE TABLES FIT
06027 002464         JMP     @SAVE3   ;NO-TOO LARGE
06030 102400         SUB     0, 0     ;OK-CLEAR AC0
06031 040473         STA     0, BN    ;INITIALIZE BYTE NUMBER
06032 040473         STA     0, SBC   ;  AND SORTED BYTE COUNT
06033 020461         LDA     0, SA    ;GET 1ST BYTE ADDRESS
06034 024461         LDA     1, FC    ;GET FIELD COUNT
06035 030461         LDA     2, MFBC  ;GET MAJOR FIELD BYTE COUNT
06036 004501         JSR     ADGEN    ;JUMP TO ADGEN
06037 020465         LDA     0, BN    ;GET BYTE NUMBER
06040 024455         LDA     1, FC    ;GET FIELD COUNT


06041 030461 MLP  :  LDA     2, SBA   ;GET PRESENT SBA
06042 050464         STA     2, SAP   ;STORE AT SORT ADDRESS POINTER
06043 004505         JSR     LOW      ;FIND LOW BYTES&PUT THEM 1ST
06044 034453         LDA     3, SFBC  ;GET SFBC
06045 116414         SUB#    0, 3, SZR ;DOES CURRENT BN=SFBC?
06046 000421         JMP     P        ;NO
06047 034453 R    :  LDA     3, SBA   ;YES-GET SBA
06050 137000         ADD     1, 3     ;ADD SBA+CNT
06051 054451         STA     3, SBA   ;STORE SUM AT SBA
06052 034453         LDA     3, SBC   ;GET SBC
06053 137000         ADD     1, 3     ;ADD SBC+CNT
06054 054451         STA     3, SBC   ;AND STORE IT AT SBC
06055 101004         MOV     0, 0, SZR ;IS BN=0?
06056 000420         JMP     Q        ;NO
06057 024436         LDA     1, FC    ;YES-GET FIELD COUNT
06060 166655         SUBOR#  3, 1, SNR ;IS FC-SBC<=1?
06061 000427         JMP     EXIT     ;YES-FINISHED!
06062 166400         SUB     3, 1     ;NO-SET FC=FC-SBC
06063 044432         STA     1, FC    ;STORE REMAINING FIELD COUNT
06064 176400         SUB     3, 3     ;CLEAR
06065 054440         STA     3, SBC   ;    SORTED BYTE COUNT
06066 000753         JMP     MLP      ;GO AGAIN
06067 125235 P    :  MOVZR#  1, 1, SNR ;IS CNT=1
06070 000757         JMP     R        ;YES-SAME AS BN=SFBC
```

```
06071 034427          LDA     3,XAPTB ;NO-GET SAP TABLE ADDRESS
06072 117000          ADD     0,3     ;ADD XAPTB+BN
06073 051400          STA     2,0,3   ;STORE SAP IN TABLE
06074 101400          INC     0,0     ;INCREMENT BN
06075 000744          JMP     MLP     ;GO AGAIN
06076 034422 @     :  LDA     3,XAPTB ;GET POINTER TABLE ADDRESS
06077 117000          ADD     0,3     ;ADD XAPTB+BN
06100 025777          LDA     1,-1,3  ;GET PREVIOUS SAP
06101 146404          SUB     2,1,SZR ;CNT=PREVIOUS SAP-PRESENT SAP
06102 125235          MOVZR#  1,1,SNR ;IS CNT<=1?
06103 100401          NEG     0,0,SKP ;YES- BN=BN-1 BY NEG/COM
06104 000735          JMP     MLP     ;NO-GO AGAIN
06105 100000          COM     0,0     ;COMPLETE NEG/COM
06106 031777          LDA     2,-1,3  ;GET PREVIOUS SAP
06107 000740          JMP     R       ;TRY AGAIN AT R

06110 020411 EXIT :   LDA     0,SBAP  ;GET SORTED BYTE ADDRESS TABLE
06111 034402          LDA     3,SAVE3 ;GET RETURN
06112 001401          JMP     1,3     ;NORMAL RETURN AT CALL+1
                      ;PROGRAM CONSTANTS&VARIABLES

06113 000000 SAVE3:   .-.
06114 000000 SA    :  .-.
06115 000000 FC    :  .-.
06116 000000 MFBC  :  .-.
06117 000000 SFBC  :  .-.
06120 006233 XAPTB:   TBLAP
06121 000000 SBAP  :  .-.
06122 000000 SBA   :  .-.
06123 007300 MAX   :  7300 7400
06124 000000 BN    :  .-.
06125 000000 SBC   :  .-.
06126 000000 SAP   :  .-.
06127 000000 RTRN  :  .-.
06130 000000 FCL   :  .-.
06131 000000 CNT   :  .-.
06132 000377 BMASK:   377
06133 000000 AP    :  .-.
06134 000000 LAP   :  .-.
06135 000000 SCNT  :  .-.
06136 000000 S3    :  .-.

                      ;ADGEN
                      ;ROUTINE TO PUT 1ST BYTE ADDRESS
                      ;OF EACH SORT FIELD INTO TABLE.
                      ;CALLING SEQUENCE-
                      ;   AC0-1ST BYTE ADDRESS OF 1ST FIELD
                      ;   AC1-FIELD COUNT
                      ;   AC2-MAJOR FIELD BYTE COUNT

06137 054770 ADGEN:   STA     3,RTRN  ;SAVE RETURN
06140 034761          LDA     3,SBAP  ;GET TABLE ADDRESS
06141 124400          NEG     1,1     ;NEGATE FIELD COUNT
06142 041400 LPA   :  STA     0,0,3   ;STORE BYTE ADDRESS
06143 125405          INC     1,1,SNR ;BUMP COUNT-SKIP IF NOT DONE
06144 002763          JMP     @RTRN   ;DONE-RETURN
06145 143000          ADD     2,0     ;NOT DONE-NEXT BYTE ADDRESS
06146 175400          INC     3,3     ;BUMP TABLE ADDRESS
06147 000773          JMP     LPA     ;GO AGAIN
                      ;LOW
```

```
                        ;FINDS LOWEST ABSOLUTE & BIT VALUE
                        ;FOUND IN ADDRESS TABLE, THE NUMBER
                        ;OF TIMES FOUND(CNT), AND THE LAST
                        ;ADDRESS OF A FOUND LOW VALUE.
                        ;THE ROUTINE THEN WORKS FROM THAT
                        ;LAST ADDRESS AND SWAPS ADDRESSES
                        ;FROM THE FIRST UNTIL THE LOW VALUE
                        ;ADDRESSES ARE ALL AT THE FIRST OF
                        ;THE TABLE.
                        ;CALLING SEQUENCE-
                        ;   AC0-BN(BYTE NUMBER IN SORT FIELD)
                        ;   AC1-FC(REMAINING FIELD COUNT)
                        ;   AC2-SBA(CURRENT LOWEST TABLE ADDRESS)
                        ;
                        ;RETURN-
                        ;AC0-BN(UNCHANGED)
                        ;   AC1-CNT(NUMBER OF LOW BYTES)
                        ;   AC2-AP(LAST LOW ADDRESS POINTER)

06150 054757 LOW   :    STA     3, RTRN   ;SAVE RETURN
06151 044757            STA     1, FCL    ;SAVE FIELD COUNT
06152 126400            SUB     1, 1      ;SET
06153 044756            STA     1, CNT    ;    CNT=0
06154 024756            LDA     1, BMASK  ;GET MAX BYTE VALUE
06155 050756            STA     2, AP     ;STORE ADDRESS POINTER
06156 032755 A     :    LDA     2, @AP    ;GET BYTE ADDRESS
06157 113000            ADD     0, 2      ;ADD BN TO GET CORRECT BYTE
06160 004443            JSR     GBYTE     ;GET BYTE
06161 146433            SUBZ#   2, 1, SNC ;IS AC2<=AC1?
06162 000411            JMP     X         ;NO
06163 146033            ADCZ#   2, 1, SNC ;YES-IS AC2<AC1?
06164 000404            JMP     Y         ;NO
06165 145000            MOV     2, 1      ;YES-NEW LOW BYTE
06166 152400            SUB     2, 2      ;RE-INITIALIZE
06167 050742            STA     2, CNT    ;     CNT TO ZERO
06170 010741 Y     :    ISZ     CNT       ;CNT=CNT+1
06171 030742            LDA     2, AP     ;UPDATE
06172 050742            STA     2, LAP    ;     LOW ADDRESS POINTER
06173 014735 X     :    DSZ     FCL       ;FCL=FCL-1  SKIP IF DONE
06174 000402            JMP     B         ;NOT DONE
06175 000403            JMP     DONE      ;DONE
06176 010735 B     :    ISZ     AP        ;AP=AP+1
06177 000757            JMP     A         ;GO AGAIN
06200 030731 DONE  :    LDA     2, CNT    ;GET LOW BYTE COUNT
06201 050734            STA     2, SCNT   ;STORE IT FOR SORT LOOP
06202 000407            JMP     Z         ;GO SWAP 1ST ADDRESS
06203 014731 DAP   :    DSZ     LAP       ;DECREMENT LAP
06204 032730 LOOP  :    LDA     2, @LAP   ;GET BYTE ADDRESS
06205 113000            ADD     0, 2      ;TO GET CORRECT BYTE
06206 004415            JSR     GBYTE     ;GET BYTE
06207 146433            SUBZ#   2, 1, SNC ;IS AC2<=AC1?
06210 000773            JMP     DAP       ;NO
06211 036715 Z     :    LDA     3, @SAP   ;YES-GET LOWEST TABLE ADDRESS
06212 032722            LDA     2, @LAP   ;GET LAST LOW ADDRESS
06213 052713            STA     2, @SAP   ;PUT LOW ADDRESS IN LOW TABLE
06214 056720            STA     3, @LAP   ;COMPLETE ADDRESS SWAP
06215 010711            ISZ     SAP       ;BUMP LOWEST ADDRESS
06216 014717            DSZ     SCNT      ;SEE IF DONE
06217 000765            JMP     LOOP      ;NOT DONE-CONTINUE
06220 024711            LDA     1, CNT    ;DONE-GET CNT
```

```
06221 030705            LDA     2,SAP   ;GET SAP
06222 002705            JMP     @RTRN   ;RETURN TO MAIN LOOP
                        ;GBYTE
                        ;ROUTINE TO GET AN 8 BIT BYTE
                        ;GIVEN THE ADDRESS
                        ;IN BYTE POINTER FORMAT
                        ;BITS 0-14------WORD ADDRESS
                        ;BIT  15--------BYTE POINTER(BP)
                        ;WHERE BP=0 POINTS TO LEFT BYTE
                        ;      BP=1 POINTS TO RIGHT BYTE
                        ;
                        ;CALLING SEQUENCE-
                        ;  AC2-BYTE ADDRESS
                        ;
                        ;RETURN-
                        ;  AC2-BYTE
                        ;AC0&AC1 UNDISTURBED
                        ;AC2&AC3 LOST

06223 054713 GBYTE:     STA     3,S3    ;SAVE RETURN
06224 151220            MOVZR   2,2     ;ORIENT WORD ADDRESS
06225 031000            LDA     2,0,2   ;GET WORD
06226 151302            MOVS    2,2,SZC ;ASSUME LEFT BYTE
06227 151300            MOVS    2,2     ;NO-RIGHT BYTE
06230 034702            LDA     3,BMASK ;OK-GET BYTE MASK
06231 173400            AND     3,2     ;BITS 8-15 LEFT
06232 002704            JMP     @S3     ;RETURN

06233 000000 TBLAP:     .-.

                        .END
```


CO
I/O

USER'S GROUP LIBRARY
PROGRAM #17 INPUT/OUTPUT PACKAGE
DATE SUBMITTED: AUGUST, 1971
DR. CHARLES S. COX
UNIVERSITY OF CAL., SAN DIEGO

A) ABSTRACT: THIS IS A PACKAGE OF TEN SUBROUTINES TO FACILITATE I/O OPERATIONS WITH THE TELETYPE. SUBROUTINE STARTING ADDRESSES AND DESCRIPTIONS ARE AS FOLLOWS.

ADDRESS	NAME	DESCRIPTION
6615	IN	RECEIVES INPUT CHARACTERS FROM THE TTY THEN CALLS OUT.
6621	OUT	OUTPUTS CHARACTER TO TTY.
7336	SP1	MAKES N SPACES ON TTY.
6631	CRLF	MAKES CARRIAGE-RETURN, LINE-FEED ON TTY.
6640	PRT	PRINTS A CHARACTER STRING ON TTY FROM LOCAL OR REMOTE STORAGE.
6725	RDTT	READS A CHARACTER STRING FROM TTY INTO STORAGE.
6405	BRK	READR NUMERALS THROUGH RDNUM AND BRANCHES ON BREAK CHARACTER ENDING THE NUMERAL STRING.
6452	RDNUM	READS A SIGNED NUMERAL STRING FROM TTY AND CONVERTS FROM OCTAL OR DECIMAL, SINGLE OR DOUBLE PRECISION INTO BINARY.
7113	TABPR	PRINTS ON TTY (USING BINU) A TABLE OF NUMBERS WITH CONTROL OF FORMAT.
204	BINU	CONVERTS BINARY NUMBER IN AC1 INTO SIGNED OR UNSIGNED, DECIMAL OR OCTAL, ASCII CHARACTERS. PRINTS RESULT ON TTY.

B) DESCRIPTIONS OF I/O PKG

IN: ACCEPTS A CHARACTER FROM TTY INTO AC0 THEN CALLS OUT.

CALL: JSR IN
<RETURN>*

OUT: TRANSMITS BITS 9-15 IN AC0 TO TTY. SETS BIT 8 TO ZERO.

CALL: JSR OUT
<RETURN>*

SP1: MAKES N SPACES ON TTY. ENTER WITH N IN AC1. AC2 AND CARRY UNCHANGED.

CALL: JSR SP1 ;WITH N IN AC1
<RETURN>*

CRLF: TRANSMITS CARRIAGE RETURN AND LINE FEED TO TTY BY USE OF OUT.

CALL: JSR CRLF
<RETURN>*

*AC0, AC1, AC2, AND CARRY ARE UNALTERED

BRK:

THIS SUBROUTINE IR USED TO INPUT NUMERICAL DATA TO THE COMPUTER AND TO BRANCH THE PROGRAM ACCORDING TO A TTY CHARACTER (THE "BREAK CHARACTER") WHICH TERMINATES THE INPUT DATA STRING.

INPUT DIGITS FROM THE TTY ARE CONVERTED INTO EQUIVALENT BINARY BY SUBROUTINE RONUM (O.V.). THEY CAN BE PREDEEDED BY A PLUS OR MINUS SIGN. THE TYPE OF CONVERSION IS CONTROLLED BY THE TWO BIT CODE CONTAINED IN AC2 UPON ENTRY TO BRK, AS FOLLOWS:

VALUE OF BIT	BIT 14	BIT 15
0	SINGLE PRECISION	DECIMAL
1	DOUBLE PRECISION	OCTAL

THE BREAK CHARACTER IS COMPARED WITH A LIST OF POSSIBLE CHARACTERS CONTAINED IN THE CALLING SEQUENCE. THE LOCATION OF THE RETURN FROM BRK IS GOVERNED BY WHICH, IF ANY, OF THE LISTED CHARACTERS IS THE SAME AS THE BREAK CHARACTER.

UPON EXIT THE NUMERICAL DATA IS IN AC0 (IF SINGLE PRECISION) OR IN AC0(LOW ORDER) AND AC1(HIGH ORDER) IF DOUBLE PRECISION. AC1 WILL BE SAVED AND RESTORED IF CONVERSION IS SINGLE PRECISION.

```
CALL:   JSR     BRK     ;NOTE 1
        .TXT    "ANY LIST OF CHARACTERS"        ;NOTE 2
        <RETURN IF ALL COMPARISONS ARE UNSUCCESSFUL>
        <RETURN IF BREAK CHARACTER IS FIRST OF LIST>
        "
        "
        "
        <RETURN IF BREAK CHARACTER IS LAST OF LIST>  ;NOTE 3
```

NOTE 1. CONVERSION TYPE IS CONTAINED IN AC2 ON ENTRY. SEE TABLE ABOVE.

NOTE 2. THE COMPARISON CHARACTERS IN THE LIST CAN BE ANY SEVEN BIT ASCII CHARACTERS EXCEPT THAT THEY MUST NOT BE ANY OCTAL DIGIT IF CONVERSION IS FROM OCTAL, OR ANY DECIMAL DIGIT IF CONVERSION IS FROM DECIMAL. THERE CAN BE ANY NUMBER OF CHARACTERS.

NOTE 3. AC1 WILL BE RESTORED IF RESULT IS SINGLE PRECISION. NUMERICAL DATA WILL BE AC0, OR AC0 (LO) AND AC1 (HI). THE OUTPUT WILL BE SIGNED ACCORDHNG TO INPUT DATA.

RDNUM:

THIS SUBROUTINE ACCEPTS AND ECHOES A SIGNED OR UNSIGNED STRING OF DIGITS FROM THE TTY AND INTERPRETS IT AS DECIMAL OR OCTAL, SINGLE OR DOUBLE PRECISION. CONVERSION TO BINARY IS UNDER THE CONTROL OF THE CONTENTS OF AC2 ON ENTRY (SEE BRK). THE RESULTANT BINARY NUMBER WILL BE EQUIVALENT TO THE INPUT DIGITAL STRING MODULO 2**15-1 (SINGLE PRECISION) OR 2**31-1 (DOUBLE PRECISION). EXIT FROM THE SUBROUTINE OCCURS WHEN A BREAK CHARACTER (SEE BRK) IS TRANSMITTED FROM THE TTY. IF THE BREAK CHARACTER IS A CARRIAGE RETURN (15) THE SUBROUTINE NOT ONLY ECHOES IT BUT ALSO TRANSMITS A LINE FEED (12) TO THE TTY. UPON EXIT THE NUMERICAL RESULT WILL BE IN AC0 (SINGLE PRECISION) OR AC0 AND AC1 (DOUBLE PRECISION, LOW ORDER IN AC0), AND THE BREAK CHARACTER WILL BE IN AC2. AC1 WILL BE SAVED AND RESTORED IF CONVERSION IS SINGLE PRECISION.

```
CALL:   JSR     BRK     ;CONVERSION TYPE IN AC2
        <RETURN>        ;RESULT IN AC0 (SINGLE PRECISION)
                        ;OR AC0 AND AC1 (DOUBLE PRECISION,
                        ;AC1 RESTORED IF SINGLE PRECISION.
```

RDTT:

READS ASCII CHARACTERS FROM TTY AND STORE THEM (2 PER WORD) IN A TEXT BUFFER IN MEMORY. FIRST WORD STORES FIRST CHARACTER IN BITS 8-15, SECOND IN BITS 0-7 AND SO ON. THIS IS ORDER REQUIRED FOR PRT. A NULL CHARACTER IS AUTOMATICALLY INSERTED AT THE END OF THE TEXT. CHARACTERS CAN BE REMOVED FROM THE BUFFER BY STRIKING THE RUBOUT KEY ON THE TTY. IF IT IS STRUCK N TIMES, N CHARACTERS ARE REMOVED. EACH CHARACTER IS PRINTED BY TTY AS IT IS ERASED.

MULTIPLE CHARACTERS CAN BE COMPACTLY ELBEDDED IN THE TEXT BUFFER IN A WAY RECOGNIZABLE BY PRT. THE METHOD IS TO STRIKE CONTROL AND A KEYS SIMULTANEOUSLY (^A) THEN STRIKE THE REPEATING CHARACTER FOR THE REQUIRED NUMBER OF TIMES. THE INFORMATION IS CODED INTN THE TEXT BUFFER AS DESCRIBED IN PRT.

IF THE NUMBER OF CHARACTERS TO BE STORED WILL EXCEED THE AVAILABLE SPACE THE TTY WILL PRINT "?" THEN MAKE A CR/LF AND WAIT FOR A NEW TEXT STRING.

RETURN FROM THE SUBROUTINE IS FORCED BY STRIKING THE ESC KEY ON THE TTY.

NOTES: IF THE OPERATOR WISHES TO ERASE A PACKED MULTIPLE CHARACTER STRING HE MUST USE ENOUGH "RUBOUTS" TO REMOVE THE CHARACTER BEFORE ^A. SUBROUTINE MAY NOT RECOGNIZE ESC OR RUBOUT IF THEY ARE THE FIRST CHARACTERS AFTER A PACKED REPEATED CHARACTER. IT MAY BE NECESSARY TO STRIKE ANOTHER ESC FOR RUBOUT BEFORE THE ARE RECOGNIZED.

ACS 0-2 ARE SAVED AND RESTORED.

```
CALL:   JSR     RDTT
        <ADDR>          ;LOWEST ADDRESS OF TEXT BUFFER
        <LTB>           ;LENGTH OF TEXT BUFFER. 2*LTB-1 IS THE
                        ;MAXIMUM NUMBER OF CHARACTERS
        0               ;ACTUAL NUMBER OF CHARACTERS WILL BE
                        ;PLACED HERE.
        <RETURN>
```

PRT:

SUBROUTINE TO PRINT A TEXT STRING OF SEVEN BIT ARCII CHARACTERS ON THE TELETYPE. PRINTINF CONTINUES UNTIL A NULL CHARACTER IS REACHED, THEN CONTROL RETURNS TO THE CALLING PROGRAM.

IF <AC0>=0 ON ENTRY, THE SUBROUTINE WILL INTERPRET THE CONTENTS OF THE FIRST LOCATION BEYOND THE JSR INSTRUCTION OF THE CALLING SEQUENCE TO BE THE LOWEST ADDRESS OF THE TEXT BUFFER. IN THIS REMOTE MODE THE TEXT CAN BE ANYWHERE IN MEMORY. RETURN WILL BE TO THE SECOND LOCATION PAST THE JSR. IN THIS LOCAL NODE THE TEXT IS ORDINARILY ASSEMBLED WITH THE PROGRAM USING THE .TXT PSEUDO INSTRUCTION. RETURN NOW OCCURS TO THE LOCATION FOLLOWING THE TEXT BUFFER.

IF, AND ONLY IF, CARRY IS ZERO ON ENTRY THE PROGRAM WILL MAKE A CR/LF BEFORE PRINTING.

THERE IS PROVISION FOR MAKING A SERIES OF N REPEATED CAHARACTERS ON THE TTY WITHOUT HAVING ALL THE CHARACTERS STORED IN THE TEXT BUFFER. SUCH PACKING IS CODED INTO THE TEXT BUFFER BY THREE SUCCESSIVE BYTES: 001, <N>, <R> (THE CHARACTER 001 CAN BE GENERATED BY ^A ON THE TTY). HERE <R> IS THE CHARACTER TO BE REPEATED N TIMES.

ACS 0-2 ARE SAVED AND RESTORED.

```
CALL:   JSR     PRT     ;NOTES 1,3
        <ADDR OF REMOTE TEXT BUFFER OR .TXT "ANY TEXT">  ;NOTE 2
        <RETURN>
```

NOTES:
1. SET CARRY=0 FOR CR/LF
2. ENTER WITH AC0=0 FOR REMOTE BUFFER PRINTING.
   ENTER WITH AC0 NOT-EQUAL TO 0 FOR LOCAL BUFFER PRINTING.
3. SEE RDTT FOR ORDER OF STORAGE.

BINU:

PRINTS OUT ON TTY THE DECIMAL OR OCTAL EQUIVALENT OF THE CONTENTS OF AC1. INDEPENDENT CONTROL ARE AVAILABLE SO AS TO REGARD THE CONTENTS OF AC1 AS SIGNED (-100000 TO +77777) OR UNSIGNED (0 TO 177777), TO SUPPRESS LEADING ZEROS OF THE PRINTOUT (BUT AT LEAST ONE DIGIT WILL PRINT), AND TO VARY THE NUMBER OF DIGITS PRINTED OVER THE RANGE 1 TO 6 INCLUSIVE. ASTERISKS ARE PRINTED IF THE REQUIRED NUMBER OF CHARACTERS IS TOO LARGE. AC2 IS SAVED AND RESTORED.

```
CALL:   JSR     BINU    ;WITH NUMBER TO BE PRINTED IN AC1
        <TYPE>          ;NOTE 1
        <NUMBER OF DIGITS>  ;NOTE 2
        <RETURN>
```

NOTE 1. BITS 13-15 CONTROL THE TYPE OF CONVERSION AS FOLLOWS:

VALUE OF BIT	BIT 13	BIT 14	BIT 15
0	SIGNED	LEADING ZEROS PRINTED	DECIMAL
1	UNSIGNED	SUPPRESSES LEADING ZEROS	OCTAL

NOTE 2. NUMBER OF TTY CHARACTERS WILL BE INCREASED BY ONE TO ALLOW FOR MINUS SIGN, OR A SPACE, IF A SIGNED CONVERSION IS TO BE MADE. IT WILL BE DECREASED IF LEADING ZEROS ARE SUPPRESSED. ACCEPTABLE VALUES FOR THE NUMBER OF DIGITS IS 1-6 INCLUSIVE. ANY OTHER VALUE WILL BE ALTERED TO 6.

TABPR:

PRINTS OUT ON TTY A TABLE OF NUMBER FROM A BLOCK IN MEMORY. CONVERSION CAN BE TO DECIMAL OR OCTAL, SIGNED OR UNSIGNED. IF CARRY IS ZERO ON ENTRY THE TTY MAKES A CR/LF BEFORE PRINTING. CONVERSION TYPE AND FORMAT OF THE PRINTING IS CONTROLLED BY FOUR BYTES IN THE CALLING SEQUENCE: TYPE, NDIG, NOSP, WPL. TYPE CONTROL THE CONVERSION AS DESCRIBED MORE FULLY FOR BINU. IN BRIEF, IF TYPE =0 THE CONVERSION IS SIGNED DECIMAL AND PRINTS LEADING ZEROS. FOR UNSIGNED CONVERSION SET TYPE=4. FOR OCTAL CONVERSION SET TYPE=1, AND TO SUPPRESS LEADING ZEROS SET TYPE=2. THESE OPTIONS CAN BE CONCATENATED. ASSUMING THAT CONVERSION IS SIGNED AND THAT LEADING ZEROS ARE NOT SUPPRESSED THE PRINTED FORMAT IS AS FOLLOWS:

```
        SDD...DD        SDD...DD        SDD...DD
```

SIGN(S) IS EITHER - OR SPACE. NUMBER DIGITS = NDIG. NUMBER OF SPACES BETWEEN WORDS = NSP. NUMBER OF WORDS PER LINE = WPL.

THE NUMBER OF CHARACTERS PER LINE IS (NSP + NDIG + 1)*WPL - NSP.

UNSIGNED CONVERSION (ALSO WITHOUT SUPPRESSION OF LEADING ZEROS) HAS THE FOLLOWING FORMAT:

```
        DDD...DD        DDD...DD        DDD...DD
          NDIG    NSP
                            WPL   WORDS
```

THE NUMBER OF CHARACTERS PER LINE IS NOW (NSP+NDIG)*WPL-NSP. IF THE REQUIRED NUMBER OF CHARACTERS EXCEEDS NDIG, ASTERISKS WILL BE PRINTED.

IF LEADING ZEROS ARE TO BE SUPPRESSED THE WORDS MAY BE SHORTENED IN LENGTH. HENCE THE NUMBER OF CHARACTERS PER LINE IS INDETERMINATE.

```
CALL:   JSR     TABPR   ;IF CRY=0, TTY MAKES CR/LF BEFORE PRINTING
        <TYPE, NDIG>    ;TYPE IS IN BITS 0-7, NDIG IN BITS 8-15
        <NOSP, WPL>     ;NOSP IS IN BITS 0-7, WPL IN BITS 8-15
        <TOTAL NUMBER OF WORDS TO BE PRINTED>
        <LOWEST ADDRESS OF CORE BLOCK CONTAINING NUMBERS>
        <RETURN>
```

EXAMPLE:

```
        JSR     TABPR
        403
        1002
        <NN>
        <ADDR>
        <RETURN>
```

WILL PRINT NN WORDS IN OCTAL FROM THE BLOCK BEGINNING AT ADDR. THE FORMAT WILL BE TWO WORDS OF THREE DIGITS PLUS SIGN PER LINE WITH TWO SPACES BETWEEN THE WORDS: SDDD SDDD

c. Listing


```
        006405  .LOC 6405
06405 054436 BRK:STA 3,SB3;NTR W/ TYPE IN AC2
06406 176400 SUB 3,3;OUTP IN AC0&1
06407 054435 STA 3,NB
06410 054435 STA 3,NB1
06411 004441 JSR RDNUM
06412 040436 STA 0,SUML
06413 044436 STA 1,SUMH;TEMP STR AC1

06414 022427 NXT:LDA 0,@SB3
06415 105300 MOVS 0,1
06416 004405 JSR COMPR
06417 121000 MOV 1,0
06420 004403 JSR COMPR
06421 010422 ISZ SB3
06422 000772 JMP NXT

06423 054424 COMPR:STA 3,SC3
06424 034422 LDA 3,R8
06425 163405 AND 3,0,SNR
06426 000407 JMP PAU
06427 010415 ISZ NB
06430 142404 SUB 2,0,SZR
06431 002416 JMP @SC3
06432 020412 LDA 0,NB
06433 040412 STA 0,NB1
06434 002413 JMP @SC3

06435 020410 PAU:LDA 0,NB1
06436 034405 LDA 3,SB3
06437 117000 ADD 0,3
06440 020410 LDA 0,SUML
06441 024410 LDA 1,SUMH
06442 001401 JMP 1,3

06443 000000 SB3:0
06444 000000 NB:0
06445 000000 NB1:0
06446 000377 R8:377
06447 000000 SC3:0
06450 000000 SUML:0
06451 000000 SUMH:0
```

```
06452 054525 RDNUM:STA 3,SS3;NTR W/TYPE IN AC2
06453 102400 SUB 0,0
06454 040525 STA 0,SIGN
06455 040774 STA 0,SUMH
06456 040772 STA 0,SUML
06457 044521 STA 1,RDN1
06460 040522 STA 0,DBL
06461 151202 MOVR 2,2,SZC
06462 000406 JMP OCTAL
06463 020522 LDA 0,JDEC;DECIMAL
06464 040436 STA 0,BR
06465 020523 LDA 0,NINE
06466 040515 STA 0,ULIM
06467 000405 JMP DSN

06470 020514 OCTAL:LDA 0,JOCT
06471 040431 STA 0,BR
06472 020515 LDA 0,SEVEN
06473 040510 STA 0,ULIM

06474 151202 DSN:MOVR 2,2,SZC
06475 004430 JSR DOUBL

06476 004517 SN:JSR IN
06477 034512 LDA 3,MIN
06500 116405 SUB 0,3,SNR
06501 000422 JMP MNUS
06502 034510 LDA 3,PLUS
06503 116404 SUB 0,3,SZR
06504 000402 JMP .+2

06505 004510 DIGIT:JSR IN
06506 030500 LDA 2,ZERO
06507 034474 LDA 3,ULIM
06510 162033 ADCZ# 3,0,SNC
06511 112032 ADCZ# 0,2,SZC
06512 000433 JMP FIN
06513 142400 SUB 2,0
06514 024735 LDA 1,SUMH
06515 030733 LDA 2,SUML
06516 151120 MOVZL 2,2
06517 125100 MOVL 1,1
06520 151120 MOVZL 2,2
06521 125100 MOVL 1,1

06522 000400 BR:JMP .+0
06523 010456 MNUS:ISZ SIGN
06524 000761 JMP DIGIT

06525 102520 DOUBL:SUBZL 0,0
06526 040454 STA 0,DBL
06527 001400 JMP 0,3

06530 151120 ADOC:MOVZL 2,2
06531 125100 MOVL 1,1
06532 113022 ADDZ 0,2,SZC
06533 125400 INC 1,1
06534 050714 STA 2,SUML
06535 044714 STA 1,SUMH
```

```
06536 000747 JMP DIGIT

06537 034711 ADECM:LDA 3,SUML
06540 173022 ADDZ 3,2,SZC
06541 125400 INC 1,1
06542 034707 LDA 3,SUMH
06543 167000 ADD 3,1
06544 000764 JMP ADOC

06545 111000 FIN:MOV 0,2
06546 034446 LDA 3,CR
06547 116405 SUB 0,3,SNR
06550 000424 JMP GLF

06551 020677 G:LDA 0,SUML
06552 024677 LDA 1,SUMH
06553 014427 DSZ DBL
06554 000411 JMP SINGL
06555 125120 MOVZL 1,1;DOUBL PREC
06556 125220 MOVZR 1,1
06557 014422 DSZ SIGN
06560 002417 JMP @SS3
06561 100404 NEG 0,0,SZR
06562 124001 COM 1,1,SKP
06563 124400 NEG 1,1
06564 002413 JMP @SS3

06565 101120 SINGL:MOVZL 0,0
06566 101220 MOVZR 0,0
06567 024411 LDA 1,RDN1
06570 014411 DSZ SIGN
06571 002406 JMP @SS3
06572 100400 NEG 0,0
06573 002404 JMP @SS3

06574 020417 GLF:LDA 0,LF
06575 004424 JSR OUT
06576 000753 JMP G

06577 000000 SS3:0
06600 000000 RDN1:0
06601 000000 SIGN:0
06602 000000 DBL:0
06603 000000 ULIM:0
06604 000406 JOCT:JMP ADOC-BR,1
06605 000415 JDEC:JMP ADECM-BR,1
06606 000060 ZERO:"0
06607 000067 SEVEN:"7
06610 000071 NINE:"9
06611 000055 MIN:"-
06612 000053 PLUS:"+
06613 000012 LF:12
06614 000015 CR:15;BRK & RDNUM JULY 4 71
```

```
06615 060110 IN:NIOS TTI
06616 063610 SKPDN,TTI
06617 000777 JMP .-1
06620 060410 DIA 0,TTI

06621 063511 OUT:SKPBZ TTO
06622 000777 JMP .-1
06623 061111 DOAS 0,TTO
06624 101300 MOVS 0,0
06625 101100 MOVL 0,0
06626 101220 MOVZR 0,0
06627 101300 MOVS 0,0
06630 001400 JMP 0,3

06631 054406 CRLF:STA 3,TEM4
06632 020762 LDA 0,CR
06633 004766 JSR OUT
06634 020757 LDA 0,LF
06635 004764 JSR OUT
06636 002401 JMP @TEM4
06637 000000 TEM4:0

06640 040457 PRT: STA 0,T00;PRT 6 22 70
06641 044457 STA 1,T00+1
06642 050457 STA 2,T00+2
06643 054457 STA 3,T00+3
06644 010456 ISZ T00+3
06645 024601 LDA 1,R8
06646 171000 MOV 3,2
06647 101005 MOV 0,0,SNR
06650 031400 LDA 2,0,3
06651 101003 MOV 0,0,SNC
06652 004757 JSR CRLF
06653 102520 SUBZL 0,0
06654 040441 STA 0,SPCNT

06655 035000 LTC:LDA 3,0,2
06656 161020 MOVZ 3,0
06657 123405 AND 1,0,SNR
06660 000426 JMP RET

06661 101235 OTP:MOVZR# 0,0,SNR
06662 000414 JMP SPACE
06663 063511 SKPBZ TTO
06664 000777 JMP .-1
06665 061111 DOAS 0,TTO
06666 014427 DSZ SPCNT
06667 000774 JMP OTP+2
06670 010425 ISZ SPCNT
```

```
06671 151100 NCH:MOVL 2,2
06672 151603 INCR 2,2 SNC
06673 000762 JMP LTC
06674 161300 MOVS 3,0
06675 000762 JMP LTC+2

06676 175063 SPACE:MOVC 3,3,SNC
06677 151401 INC 2,2,SKP
06700 175301 MOVS 3,3,SKP
06701 035000 LDA 3,0,2
06702 161000 MOV 3,0
06703 123400 AND 1,0
06704 040411 STA 0,SPCNT
06705 000764 JMP NCH

06706 020411 RET:LDA 0,T00
06707 024411 LDA 1,T00+1
06710 155000 MOV 2,3
06711 030410 LDA 2,T00+2
06712 101005 MOV 0,0,SNR
06713 002407 JMP @T00+3
06714 001401 JMP 1,3

06715 000000 SPCNT:0
06716 000000 CYS:0
06717 000000 T00:0
06720 000000 0
06721 000000 0
06722 000000 0
06723 000000 0
06724 000077 QUEST:"?

06725 040772 RDTT:STA 0,T00;6 21 71
06726 044772 STA 1,T00+1
06727 050772 STA 2,T00+2
06730 054772 STA 3,T00+3
06731 021401 RDT:LDA 0,1,3
06732 040557 STA 0,WCNT
06733 031400 LDA 2,0,3
06734 175003 MOV 3,3,SNC
06735 004674 JSR CRLF
06736 126400 SUB 1,1
06737 044756 STA 1,SPCNT
06740 120000 COM 1,0
06741 040754 STA 0,SPCNT
```

```
---
↑↑↑
   06742 004653 GCH:JSR IN
   06743 034547 LDA 3,ESCP
   06744 116405 SUB 0,3,SNR
   06745 000431 JMP CAR
   06746 034536 LDA 3,RUB
   06747 116405 SUB 0,3,SNR
   06750 000447 JMP ERAS
   06751 125400 INC 1,1
   06752 101235 MOVZR# 0,0,SNR
   06753 000465 JMP NSP
   06754 125213 MOVR# 1,1,SNC
   06755 004403 JSR STW

   06756 040745 Q:STA 0,T00+4
   06757 000763 JMP GCH

   06760 175400 STW:INC 3,3
   06761 054527 STA 3,SV3
   06762 034741 LDA 3,T00+4
   06763 101300 MOVS 0,0
   06764 163000 ADD 3,0
   06765 041000 STA 0,0,2
   06766 151400 INC 2,2
   06767 014522 DSE WCNT
   06770 002520 JMP @SV3

   06771 020733 R:LDA 0,QUEST
   06772 004627 JSR OUT
   06773 004636 JSR CRLF
   06774 034726 LDA 3,T00+3
   06775 000734 JMP RDT

   06776 004633 CAR:JSR CRLF
   06777 125213 MOVR# 1,1,SNC
   07000 000407 JMP .+7
   07001 151400 INC 2,2
   07002 020721 LDA 0,T00+4
   07003 034502 LDA 3,RR8
   07004 163400 AND 3,0
   07005 041377 STA 0,-1,2
   07006 000403 JMP .+3
   07007 102400 SUB 0,0
   07010 041000 STA 0,0,2
   07011 020706 LDA 0,T00
   07012 030707 LDA 2,T00+2
   07013 034707 LDA 3,T00+3
   07014 045402 STA 1,2,3
   07015 024703 LDA 1,T00+1
   07016 001403 JMP 3,3

   07017 124400 ERAS:NEG 1,1
   07020 124000 COM 1,1
   07021 125212 MOVR# 1,1,SZC
   07022 000404 JMP LDW
   07023 020700 LDA 0,T00+4
U  07024 006350 JSR @COUTC
   07025 000715 JMP GCH
```

6557

```
07026 150400 LDW:NEG 2,2
07027 150000 COM 2,2
07030 010461 ISZ WCNT
07031 021000 LDA 0,0,2
07032 034453 LDA 3,RR8
07033 117400 AND 0,3
07034 054667 STA 3,T00+4
07035 101300 MOVS 0,0
07036 006545 JSR @COUTC
07037 000703 JMP GCH

07040 006542 NSP:JSR @CINC
07041 010654 ISZ SPCNT
07042 000403 JMP NFST
07043 040440 STA 0,RPTCH;REPEATED CHAR
07044 000774 JMP NSP

07045 034436 NFST:LDA 3,RPTCH
07046 116405 SUB 0,3,SNR
07047 000771 JMP NSP
07050 040437 STA 0,BKSTR
07051 125400 INC 1,1
07052 125400 INC 1,1
07053 125400 INC 1,1
07054 125213 MOVR# 1,1,SNC
07055 000413 JMP EVEN
07056 102520 SUBZL 0,0
07057 004702 JSR STW+1
07060 034635 LDA 3,SPCNT
07061 054642 STA 3,T00+4
07062 020421 LDA 0,RPTCH
07063 004676 JSR STW+1
07064 102000 ADC 0,0
07065 040630 STA 0,SPCNT
07066 020421 LDA 0,BKSTR
07067 000667 JMP Q

07070 176520 EVEN:SUBZL 3,3
07071 054632 STA 3,T00+4
07072 020623 LDA 0,SPCNT
07073 004666 JSR STW+1
07074 034407 LDA 3,RPTCH
07075 054626 STA 3,T00+4
07076 020411 LDA 0,BKSTR
07077 004662 JSR STW+1
07100 102000 ADC 0,0
07101 040614 STA 0,SPCNT
07102 000640 JMP GCH

07103 000000 RPTCH:0
07104 000177 RUB:177
07105 000377 RR8:377
07106 006631 CCRLF:CRLF
07107 000000 BKSTR:0
07110 000000 SV3:0
07111 000000 WCNT:0
07112 000033 ESCP:33;RDTT 6 22 71
```

```
07113 040454 STA 0,SS0;TABPR
07114 044454 STA 1,SS1
07115 050454 STA 2,SS2
07116 102000 ADC 0,0;CY UNCH
07117 040454 STA 0,FLG
07120 024765 LDA 1,RR8
07121 054451 STA 3,SSS3

07122 021400 A:LDA 0,0,3
07123 111300 MOVS 0,2
07124 123400 AND 1,0
07125 133400 AND 1,2
07126 175400 INC 3,3
07127 010444 ISZ FLG
07130 000404 JMP .+4
07131 040420 STA 0,NDIG
07132 050416 STA 2,TYPE
07133 000767 JMP A
07134 040441 STA 0,WPL
07135 050442 STA 2,NOSP
07136 021400 LDA 0,0,3
07137 040441 STA 0,NW
07140 031401 LDA 2,1,3
07141 101003 MOV 0,0,SNC

07142 006744 NEXL:JSR @CCRLF
07143 020432 LDA 0,WPL
07144 040432 STA 0,WPL1

07145 025000 NEXW:LDA 1,0,2
07146 151400 INC 2,2
07147 004435 JSR BINU
07150 000000 TYPE:0
07151 000000 NDIG:0
07152 014426 DSZ NW
07153 000406 JMP NEX
07154 020413 LDA 0,SS0;*
07155 024413 LDA 1,SS1
07156 030413 LDA 2,SS2
07157 034413 LDA 3,SSS3
07160 001404 JMP 4,3

07161 014415 NEX:DSZ WPL1
07162 000402 JMP .+2
07163 000757 JMP NEXL
07164 024413 LDA 1,NOSP
07165 006414 JSR @SPC
07166 000757 JMP NEXW
```

```
---
???
    07167 000000 SS0:0
    07170 000000 SS1:0
    07171 000000 SS2:0
    07172 000000 SSS3:0
    07173 000000 FLG:0
    07174 000000 0
    07175 000000 WPL:0
    07176 000000 WPLI:0
    07177 000000 NOSP:0
    07200 000000 NW:0
    07201 007336 SPC:SP1
    07202 006615 CINC:IN
    07203 006621 COUTC:OUT

    07204 050527 BINU:STA 2,S2;BINU
    07205 054527 STA 3,S3
    07206 031401 LDA 2,1,3;NO.DIGITS/WORD
    07207 020523 LDA 0,C6
    07210 150557 NEGOL# 2,2,SBN;IF NDIG<=0
    07211 111000 MOV 0,2;OR >6, REPLACE BY 6
    07212 040737 STA 0,NDIG
    07213 142513 SUBL# 2,0,SNC
    07214 050735 STA 2,NDIG
    07215 102520 SUBZL 0,0
    07216 101100 MOVL 0,0
    07217 031400 LDA 2,0,3
    07220 115100 MOVL 0,3
    07221 143600 ANDR 2,0
    07222 040513 STA 0,SUPZF;FLG=1 SUPR LDNG 0'S
    07223 157405 AND 2,3,SNR;SIGNED?
    07224 000417 JMP SIGN1

    07225 034460 B0:LDA 3,RADR;COMP ADR OF COPAR NTGRS&RADIX
    07226 151202 MOVR 2,2,SZC
    07227 034457 LDA 3,RADR+1
    07230 031776 LDA 2,-2,3
    07231 050472 STA 2,RDX
    07232 030717 LDA 2,NDIG
    07233 156400 SUB 2,3
    07234 054471 STA 3,BASAD;ADR OF CMPRSN NTGR

    07235 034466 B1:LDA 3,RDX
    07236 054466 STA 3,MAX
    07237 034466 LDA 3,BASAD
    07240 031400 LDA 2,0,3;GET CMPRSN NTGR

    07241 020470 A1:LDA 0,C60
    07242 000416 JMP A3

    07243 125133 SIGN1:MOVZL# 1,1,SNC
    07244 000405 JMP POS
    07245 020461 LDA 0,MINUS
    07246 124400 NEG 1,1

    07247 006734 B2:JSR @COUTC
    07250 000755 JMP B0

    07251 020457 POS:LDA 0,SP
    07252 000775 JMP B2
```

```
---
↑↑↑
   07253 146400 A2:SUB 2,1
   07254 101400 INC 0,0
   07255 014460 DSZ SUPZF
   07256 000401 JMP .+1
   07257 014445 DSZ MAX

   07260 146433 A3:SUBZ# 2,1,SNC
   07261 000402 JMP FFIN
   07262 000771 JMP A2

   07263 034441 FFIN:LDA 3,MAX
   07264 175005 MOV 3,3,SNR
   07265 020442 LDA 0,ASTR
   07266 150015 COM# 2,2,SNR
   07267 020442 LDA 0,C60
   07270 014445 DSZ SUPZF
   07271 006712 JSR @COUTC
   07272 010443 ISZ SUPZF
   07273 000401 JMP .+1
   07274 010431 ISZ BASAD
   07275 151224 MOVZR 2,2,SZR;CMPRSN NTGR=1?
   07276 000737 JMP B1;NO,GET NEXT
   07277 014436 DSZ SUPZF;YES
   07300 000402 JMP RETUR;AT LEAST ONE LDNG 0 WILL B PRNTD
   07301 006702 JSR @COUTC

   07302 030431 RETUR:LDA 2,S2
   07303 034431 LDA 3,S3
   07304 001402 JMP 2,3

   07305 007315 RADR:RDX10+2
   07306 007323 RDX8+2
   07307 177777 177777
         000012 .RDX 10
   07310 023420 10000
   07311 001750 1000
   07312 000144 100
   07313 000012 RDX10:10
   07314 000001 1
         000010 .RDX 8
   07315 100000 100000
   07316 010000 10000
   07317 001000 1000
   07320 000100 100
   07321 000010 RDX8:10
   07322 000001 1
   07323 000000 RDX:0
   07324 000000 MAX:0
   07325 000000 BASAD:0
   07326 000055 MINUS:"-
   07327 000052 ASTR:"*
   07330 000040 SP:40
   07331 000060 C60:60
   07332 000006 C6:6
```

```
---
↑↑↑
   07333 000000 S2:0
   07334 000000 S3:0
   07335 000000 SUPEF:0

   07336 124000 SP1:COM 1,1;ENTER W/ NO OF SPACES IN AC1
   07337 020771 LDA 0,SP
   07340 054406 STA 3,SS3V
   07341 125405 INC 1,1,SNR
   07342 002404 JMP @SS3V
   07343 006402 JSR @OUT1
   07344 000775 JMP .-3
   07345 006621 OUT1:OUT
   07346 000000 SS3V:0;TABPR,BINU&SP1 JULY 3 71
```

.END

A	007122
A1	007241
A2	007253
A3	007260
ADECM	006537
ADOC	006530
ASTR	007327
B0	007225
B1	007235
B2	007247
BASAD	007325
BINU	007204
BKSP	007347
BKSTR	007107
BR	006522
BRK	006405
C6	007332
C60	007331
CAR	006776
CCRLF	007106
CINC	007202
COMPR	006423
COUT	007027
COUTC	007203
CR	006614
CRLF	006631
CYS	006716
DBL	006602
DIGIT	006505
DLAY	007107
DOUBL	006525
DSN	006474
END	007436
ENDER	007471
EOFB	007503
ERAS	007017
ES	007468
ES1	007520
ESCP	007112
EVEN	007070
FFIN	007263
FIN	006545
FLG	007173
FWSP	007365
G	006551
GCH	006742
GLF	006574
GPIR	007364
GPPI	007405
IN	006615
JDEC	006605
JOCT	006604
L	007533
LDW	007026
LF	006613
LTC	006655
MAX	007324
MIN	006611
MINUS	007326
MNUS	006523

N	007502
NB	006444
NB1	006445
NCH	006671
NDIG	007151
NE	007423
NEX	007161
NEXL	007142
NEXW	007145
NFG	007406
NFST	007045
NINE	006610
NLD	007541
NOSP	007177
NSP	007040
NW	007200
NXT	006414
OCTAL	006470
OTP	006661
OUT	006621
OUT1	007345
PAR	007503
PAU	006435
PI	007431
PIC	007505
PLUS	006612
POS	007251
PR	007476
PRT	006640
Q	006756
QUEST	006724
R	006771
R8	006446
RADR	007305
RDMTP	007410
RDN1	006600
RDNUM	006452
RDT	006731
RDTT	006725
RDX	007323
RDX10	007313
RDX8	007321
RET	006706
RETUR	007302
RPTCH	007103
RR8	007105
RRWLD	007456
RUB	007104
RWLD	007534
S2	007333
S3	007334
SB3	006443
SC3	006447
SEVEN	006607
SIGN	006601
SIGN1	007243
SINGL	006565
SN	006476
SP	007330
SP1	007336

SPACE	006676
SPC	007201
SPCNT	006715
SS0	007167
SS1	007170
SS2	007171
SS3	006577
SS3V	007346
SSS3	007172
STW	006760
SUMH	006451
SUML	006450
SUPZF	007335
SV3	007110
TEM4	006637
TC0	006717
TYPE	007150
ULIM	006603
WCNT	007111
WPL	007175
WPL1	007176
WRMT	007504
WRWLD	007535
ZERO	006606
.MWS	007477

D) MINIMUM HARDWARE REQUIREMENTS: NOVA + 4K MEMORY

E) TOTAL MEMORY REQUIREMENTS: 742 (OCTAL) LOCATIONS

F) NO OTHER SUBROUTINES ARE REQUIRED

G-H) I/O PKG IS WRITTEN IN ASSEMBLY LANGUAGE IN LOCATIONS 6403-7346. IT CAN BE REASSEMBLED INTO ANY OTHER LOCATION. IT REQUIRES 742 LOCATIONS.

Wire List Layout Program
Program #C19
By: R. A. Farwell
Ottawa, Ontario, Canada
September 27, 1971

INTRODUCTION

ABSTRACT

The wire list generation system is designed to allow the user to quickly layout the wiring list for a group of printed circuit cards that have wire wrap interconnections. The existing system allows the wiring list for eight 86 pin cards to be generated. Each pin is described by six ASCII characters. The program has nine commands:

BLANKS	-	Blank the storage area
ENTER	-	Enter data thrcugh the teletype
READ	-	Read data from the high speed reader
CHANGE	-	Change data from the teletype
DELETE	-	Delete a card entry from the table
LIST	-	List all card data on the teletype
LIST1	-	List one card data on the teletype
PUNCH	-	Punch data on the high speed punch
WIRE	-	Layout an optimized wiring diagram

This document describes the operation of a program referred to as "WIRE". It explains how "WIRE" can be used to make up the required wiring lists for any interconnections between a group of printed circuit cards.

At present the program is configurated to handle eight double sided printed circuit cards containing up to forty-three pins per side. Pin designations are by number on one side and by letter on the other. The numbering is continuous from one to forty-three, inclusive, while the lettering goes from A to YY omitting the following letters:- G,I, O,Q,GG,II,OO, and QQ.

SYSTEM GENERATION

The WIRE program is supplied to the user as a five source tapes and a binary tape. It operates in 4096 words of core memory using all locations up to 7630(8).

As the program exists it is for use with a model 35 teletype and a high speed paper tape reader and punch unit. If one or more of these devices is not present the user will have to make modifications to the source tapes.

If the model 35 teletype was not present the user would have to modify the ".TYPE" subroutine of tape 2 to maintain a line counter. This will allow simulation of the form feed character that is typed out by subroutine ".TYP3". Similar modifications could be made to implement the line printer as the listing device.

For a system without the high speed paper tape equipment only minor modifications are necessary. To use the paper tape punch on the teletype the subroutine "PUN" on tape five should be changed to

```
PUN:    SKPBZ  TTO
        JMP    .-1
        DOAS   Ø,TTO
        JMP    Ø,3
```

To use the high speed reader the user need only leave data switch zero in the OFF position when using the WIRE program READ command.

PROGRAM OPERATION

The binary tape should be loaded into core memory using the Binary Loader. This tape has a start block at the end so the program will start immediately upon completion of the loading.

The program may be restarted at any time by setting all the console data switches to zero, all off, and press RESET and then START. Stored at location Ø is a jump indirect through location 40 which contains the starting address of the program.

COMMAND DECODER

At the start of the program and after each command is completed, a branch is made to the Command Decoder. This routine will ask the user to enter the next command by typing out a carriage return, line feed, and "CMND-". The user should then enter a command which can be up to six characters in length. There are nine possible commands that can be used ranging in length from four to six characters.

The command decoder will accept up to six characters but if only four or five are required it may be terminated by typing a carriage return or by entering spaces for the remaining characters. The commands that are acceptable and their usage is given below.

BLANK	-	BLANK THE STORAGE AREA
ENTER	-	ENTER DATA THROUGH THE TELETYPE
READ	-	READ DATA FROM THE HIGH SPEED READER
CHANGE	-	CHANGE DATA FROM TELETYPE
DELETE	-	DELETE A CARD ENTRY FROM THE TABLE
LIST1	-	LIST ONE TABLE ON THE TELETYPE
LIST	-	LIST DATA ON THE TELETYPE
PUNCH	-	PUNCH DATA ON THE HIGH SPEED PUNCH
WIRE	-	LAYOUT THE WIRING DIAGRAM

A. ENTER Command

This section of the program first checks to make sure there is room for another card entry in the storage table. It should be remembered that as the present system exists only eight card entries are allowed. If these are all used at the time the ENTER command is issued, an error message "NO ROOM" will be typed and the Decoder will request a new command. At this time the user could use the DELETE command to remove tables that are not being used.

After successfully finding a card entry table that is not used, the program will request the user to enter the table title by typing a carriage return, line feed, and "NAME-". A twenty character title is allowed for each card. Other sections of the WIRE program will identify it by the first four characters but the user may want to

put a longer description than what four characters allows. A couple of examples follow:

MDM MODEM DEC. 11/7Ø
CIMI OCTOBER 25,197Ø

The only restriction on the title to be entered is that the first character must be alphabetic between A and Z. If this rule is not adhered to the program will request the name again. If the user gives a name of less than four characters, spaces are substituted for the remaining characters.

After the title is entered a check is made to determine whether the first four characters spell "DONE". If the answer is yes, a branch is made immediately back to the Command Decoder.

When the title passes this check, the program requests pin configuration data. There is no restriction to the order in which the pin data is entered. A pin description can be changed by re-entering the same pin designation and its corresponding data. The last pin description entered is always the one that resides in the table so that any mistake on entry can be corrected by re-entry.

Pin data is requested by the program typing a carriage return, line feed, and ":-". The user must then enter a two character pin name. If the name is only one character in length like the numbers 1 - 9 or alphabetic characters A - Z, the user must also enter either a leading or trailing space character to complete the name.

If it was not found to be a legal pin name the program will reply with " PER-" and then re-issue the request for the pin name as described above. If the pin name entered was two space characters or a carriage return the program will assume that the user has completed the data entry for this table and it will branch back to the Command Decoder.

After a legal pin name has been entered the program will type a single space and then wait for the user to enter a single pin direction character. This character can designate an input to the card (I) or an output from the card (O).

Only one other character can legally be entered as the direction character. This is a "D" which means that the previous description for this pin is to be deleted completely. The pin area in the table is blanked out and the next pin description is requested.

If any other character other than an "I", "O", or "D" is entered, it is diagnosed as an error and a message " DER-" is typed out and the program waits for the user to input the pin direction again.

After the signal direction has been successfully entered, the program will type a single space character and wait for the user to enter a six character signal name. The name can be terminated before all six characters are inputted by the user typing a carriage return. Any

character string can be used as the signal name but it is desireable that the name have printing characters only. If the name string is terminated before the full six characters are entered, blanks will be substituted for the remaining characters.

The program then re-cycles back and requests further pin descriptions to be entered. This cycle continues until the user terminates it by the method mentioned above (double spaces or a carriage return).

An example of the form the data input would take is listed below.

```
CMND--ENTER
NAME-MDM OCTOBER 25/70
:-1 I  5V
:-A I  5V
:-28 I GND
:-FF I GND
:-D O EOT/
:-6 O MDMKEY
:-F O MDMKEY
:-14 I  12V
:-R I  12V
:-L O CTS/
:-S I CTS/
:-17 I EOT
:-23 I CTS
:-5 I CTS
:-Y I EOT
:-22 I SEND/
:-16 O TE
:-
CMND-LIST
```

B. CHANGE Command

The program first requests the name of the card whose pin configuration data is to be changed. The user can enter up to a twenty character name but only the first four will be used for determining if this card is present in the storage table. If the card cannot be found the program will type out "NAME?" and request the name to be entered again.

Once the name of a card description within the storage table is found the program will operate the same as for the ENTER command.

When the user is finished making changes for the present card, he exits by typing a carriage return or two spaces for the pin number similar to the ENTER command. The system will then request the name of the next card to be changed. If no further modifications are necessary the user can enter "DONE" to indicate the CHANGE operation has been completed.

C. DELETE Command

After successfully entering "DELETE" to the Command Decoder the program will request the name of the table to be deleted. If the name entered does not correspond to one of the table names, the error message "NAME?" is typed and the table name is requested again.

Once the name of an existing table has been entered, the table is deleted and the other tables are compressed. After the name of the next table to be deleted. If that was the last table, the user should type "DONE".

D. LIST1 Command

This command causes the program to output a listing of a single table onto the teletype. It first requests the name of the table to be listed and checks whether the table is present. If it is found to be present the program branches to the top of a new page, types out the card title, and then a listing of all pins used on that card.

An example of the command string and the resulting listing is given below:

```
CMND-LIST1
NAME-MDM

MDM OCTOBER 25/70

PIN   DIR  NAME     PIN   DIR   NAME

 1     I    5V      A     I      5V
 5     I   CTS/     E
 6     O   MDMKEY   F     O     MDMKEY
10                  L     O     CTS/
14     I    12V     R     I      12V
15                  S     I     CTS/
16     O   TE       T
17     I   EOT      U
18                  V     O     TE
21                  Y     I     EOT
22     I   SEND/    Z
23     I   CTS      AA
28     I   GND      FF    I     GND
```

E. LIST Command

The list command is similar to the LIST1 command in the form of output produced but it lists the data for all existing card tables in storage. Each card listing is started on a new page. A condensed version of a typical listing follows.

```
TRM OCTOBER 25/70

PIN   DIR  NAME     PIN   DIR   NAME

 1     I    5V      A     I      5V
 5     O   PCE/     E
10     I   RE       L
11     I   PE       M     I     TL/
12     O   PED/     N
16                  T     I     TSS/
18                  V     I     TS
19     I   TE       W     I     START/
22     I   RSS/     Z
43     I   GND      YY    I     GND
```

TM OCTOBER 25/70

PIN	DIR	NAME	PIN	DIR	NAME
1	I	5V	A	I	5V
3	O	EOT	C		
4	O	PE/	D	O	EOT/
5			E	O	PE
7	O	TSS/	H		
8	O	TS	J		
16	O	TMKEY	T	O	TMKEY
17	I	PED/	U		
18			V	O	RSS/
20			X	O	PRS
22	O	TPS	Z		
28	I	GND	FF	I	GND

MDM OCTOBER 25/70

PIN	DIR	NAME	PIN	DIR	NAME
1	I	5V	A	I	5V
4			D	O	CTS/
5	I	CTS	E		
6	O	MDMKEY	F	O	MDMKEY
14	I	_12V	R	I	12V
15			S	I	CTS/
16	O	TE	T		
17	I	EOT	U		
21			Y	I	EOT
22	I	SEND/	Z		
23	I	CTS	AA		
28	I	GND	FF	I	GND

F. PUNCH Command

This command causes the program to output on the high speed paper tape punch a binary tape corresponding to the table area currently being used.

Upon entry to the PUNCH section of the program a check is made to determine if there are any card tables defined. If there are no tables defined, an error message "NO ENTRIES" is typed out and the program branches back to the Command Decoder for the next user request.

If active tables were found the program punches out 300(8) blanks equivalent to about twenty inches of leader tape. The active data tables are then punched out ignoring any space characters that might be present. After all the tables have been punched the program punches a done block giving as the starting address the start of the Command Decoder. Another 300(8) blanks or about twenty-five inches of trailer tape is then produced. The program branches back to the Command Decoder for the next request.

G. READ Command

The data tape produced by the PUNCH command can be read into the program from the high speed paper tape reader by using the READ command. Before entering "READ" to the Command Decoder, data switch Ø on the console should be set ON (logical 1) and the paper tape loaded into the reader.

Once the "READ" command is entered the program blanks out the complete storage area and then branches to the Binary Loader to read in the data tape. The done block will cause the program to branch back to the Command Decoder when the data tape has been completely read into the table storage area.

H. WIRE Command

The operations performed when this function is requested are the most complex of all those performed by this program. It requires so much CPU time that the program frequently becomes compute bound as opposed to I/O bound.

The program produces a wiring diagram or list showing all interconnecting wires that should be placed on the mother board containing the printed circuit cards. The list can be used at equipment assembly time.

The first operation of this section of the program is to ask for a list of card tables that gives the order of wiring the cards together. This list should correspond in order to the arrangement of the cards in the physical equipment.

Since the program has been written so that up to eight different card descriptions can be contained in core at any one time, the user can specify that any number of cards up to eight can be wired together. If fewer than eight cards are to be wired together, the user must end the table name entry section by entering "GO". This instructs the program to go to the start of the wire diagram layout program and HALT. If "DONE" is entered as the name of a card table, the program will branch back to the Command Decoder for the next operation.

After the program has been given the go ahead (user entered "GO" for a card table name or has entered eight card table names), it will proceed to produce the wiring diagram. This produced wiring list is in a form that can be conveniently used by the assembly line personnel in wiring up the back panel of the equipment being manufactured.

The program will first list all interconnections for the number One card and then the interconnections between the number 1 and number 2 cards, the number 1 and number 3 cards, etc. After listing all interconnections for card number 1 the program will give a listing of any pins that are unused. By definition a pin is said to be unused if input pin does not have a signal tied to it. Unused output pins are ignored as they can be provisions for future use. After card number one is completed, a similar list is made for card number two giving connections between card 2 and card 2, card 2 and card 3, etc.

A typical wiring list for a three card assembly is given below. The

necessary input commands are first given. It should be noted that for the timing card TM that all input signals were satisfied but for the transmitter - receiver module TRM and the modulator -- demodulator module MDM there were a number of pins unused. These signals would most likely come from an external power supply and/or additional interface unit.

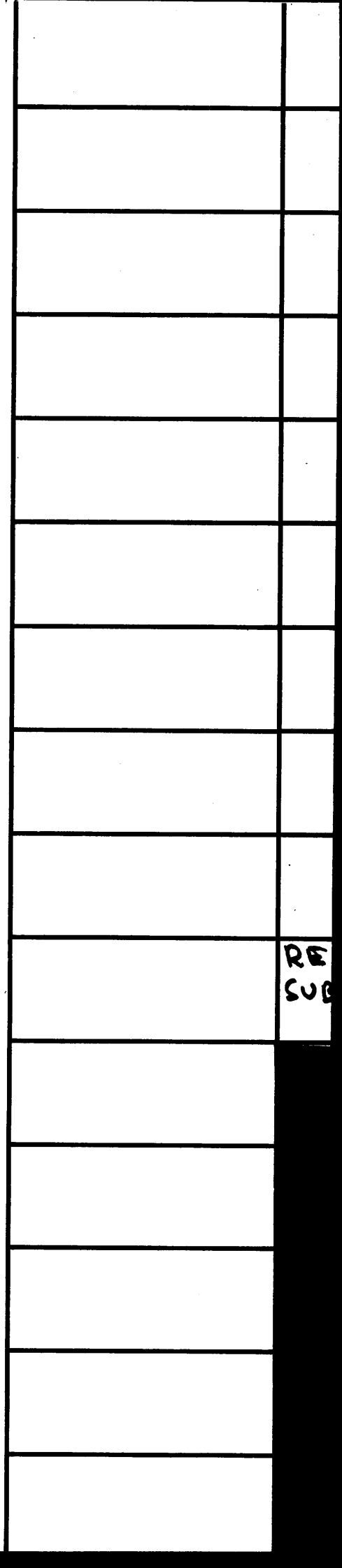
```
CMND-WIRE
NAME-TM
NAME-TRM
NAME-MDM
NAME-GO

      TM       OCTOBER 25/70

       TM        TO      IM

        1      -TO-      A     5V
       28      -TO-     FF    GND

       TM        TO     TRM

        1      -TO-      1     5V
        7      -TO-      T    TSS/
        8      -TO-      V     TS
       17      -TO-     12    PED/
       28      -TO-     43    GND
        E      -TO-     11     PE
        V      -TO-     22    RSS/

       TM        TO     MDM

        3      -TO-     17    EOT

     INPUT PINS NOT USED

    TRM       OCTOBER 25/70

     TRM        TO     TRM

        1      -TO-      A     5V
       43      -TO-     YY    GND

     TRM        TO     MDM

        1      -TO-      1     5V
       19      -TO-     16     TE
       43      -TO-     28    GND

     INPUT PINS NOT USED

      10 RE
       M TL/
       W START/
```

MDM OCTOBER 25-70

MDM	TO	MDM	
1	-TO-	A	5V
5	-TO-	D	CTS/
14	-TO-	R	12V
17	-TO-	Y	EOT
28	-TO-	FF	GND
D	-TO-	S	CTS/

INPUT PINS NOT USED

1 5V
14 12V
22 SEND/
23 CTS
28 GND


RE
SUB

Program #021
Date Submitted: 1/31/72

TEPHAN M. HEIDER
EPT. OF PHYSICS
TATE UNIVERSITY OF NEW YORK/BUFFALO
UFFALO, NEW YORK 14214

NSTRUCTIONS FOR RESLB
RELOCATABLE EXTENDED SINGLE USER BASIC)

. MINIMUM CONFIGURATION

K NOVA LINE COMPUTER MANUFACTURED BY DATA GENERAL
ITH STANDARD TELETYPE INTERFACE. IT IS URGED THAT
SERS HAVE AT LEAST 12K CORE. RESLB IS ABOUT
S ABOUT 14K(OCTAL) WORDS LONG.

. OPERATION:

ESLB MAY BE LOADED WITH THE RELOCATABLE LOADER AS
STAND ALONE PROGRAM. USER SUPPLIED MACHINE LANGUAGE
ROGRAMS SHOULD BE RELOCATABLE AND GENERALLY LOADED BEFORE
ESLB IS LOADED.
F THE PROGRAM IS LOADED WITH THE RELOCATABLE
DER, IT MAY BE STARTED BY PUSHING CONTINUE OR ELSE
T MAY BE STARTED AT LOCATION 20.
FTER STARTING BASIC A FEW QUESTIONS WILL BE PRINTED ON THE
ELETYPE. AFTER ANSWERING ALL OF THE QUESTIONS, THE
ESCAPE" KEY MUST BE PUSHED.(CNTRL SHIFT K) BASIC WILL
OW TYPE "*READY". SHOULD YOU WISH TO RESTART THE
OMPUTER, USE LOCATION 2 AS THE STARTING LOCATION.
DO NOT RESTART AT 20) EACH TIME YOU RESTART,
USH THE "ESCAPE" KEY.

S A FREE STANDING PROGRAM YOU MAY USE THE INSTRUCTIONS
OR DATA GENERAL'S TIME SHARE BASIC. (IF THEY DID NOT
END YOU INSTRUCTIONS FOR TIME SHARE BASIC, GET HOLD
F YOUR SALESMAN!)
HE TIME COMMAND WILL GIVE 0 FOR A FREE STANDING VERSION
F RESUB

O NOT WORRY ABOUT UNSATISFIED EXTERNALS WHEN
DING RESLB ALONE. THESE ARE ALL CHECKED BY THE
NCE ONLY CODE IN RESUB.
OCATIONS 0-10 ARE NOT LOADED BY THE LOADER,
UT ARE SET UP BY THE ONCE ONLY CODE. THIS SHOULD
LLOW RESUB TO BE LOADED UNDER DOS.
F THE QUESTION ABOUT HARDWARE MULTIPLY DIVIDE IS
NSWERED "Y", THE HARDWARE MULTIPLY
HOULD SPEED UP THE FLOATING POINT MULTIPLY ROUTINE.
(ORIGINAL NOVA MULTIPLY DIVIDE
NIT WILL NOT BE COMPATABLE)

. ASSEMBLY LANGUAGE PROGRAMS:

VARIETY OF ASSEMBLY LANGUAGE PROGRAMS MAY BE USED
N CONJUNCTION WITH RESUB. THE SOURCE TAPE OF A SAMPLE
ROGRAM IS SUPPLIED GIVING EXAMPLES. VARIOUS DISCUSSIONS
ILL FOLLOW AS TO THE VARIOUS PROVISIONS INCLUDED
S WELL AS SOME OF THE COMMENTS FROM THE RESLB LISTING.

; 4. BACKGROUND PROGRAMS

;THE VARIOUS BACKGROUND PROGRAMS WILL BE IDENTIFIED
;BY THE LOCATION TO WHICH RESUB WILL JUMP
;WHEN THE PROGRAM IS CALLED. THE RETURN TO RESUB
;IS ALSO GIVEN. USE OF THESE PROGRAMS IS
;OPTIONAL. THE ONCE ONLY CODE IN RESUB WILL
;CHECK ALL EXTERNALS. IF THERE ARE NO
;CORRESPONDING USER SUPPLIED ENTRY
;POINTS, A SUITABLE PATCH IS MADE. THUS, YOU NEED ONLY
;WORRY ABOUT THOSE FEATURES THAT YOU ARE GOING TO USE.
;NO "DUMMY" ENTRIES NEED BE USED WITH RESUB.
;NOTE: DO NOT USE LOCATION 3771

;WOC: (ONCE ONLY CODE) THIS IS A PROGRAM
;THAT IS USED ONLY AT THE INTIAL STARTUP.
;RETURN TO THE ADDRESS IN AC3 OR TO WOCR.

;STRUP: (STARTUP) THIS CODE IS EXECUTED EACH TIME
;THE COMPUTER IS STARTED. AT INITIAL STARTUP
;IT IS EXECUTED AFTER THE ONCE ONLY CODE.
;YOU MUST RETURN TO BASIC WITH A JMP @.TASK
;NOTE: .TASK IS IN PAGE ZERO OF RESUB

;BGNDP: (BACKGROUND PROGRAM) THIS CODE WILL BE EXECUTED
;PERIODICALLY WHEN RESUB IS NOT DOING CALCULATIONS.
;IT IS SUGGESTED THAT YOU DO NOT MAKE THIS PROGRAM
;CONSUME MORE THAN 100 MS.
;DO NOT LEAVE THE INTERRUPT SYSTEM OFF SO LONG THAT
;RESUB CANNOT KEEP UP WITH THE TELETYPE.
;RETURN IS A JMP @.TASK

;BASEV: IF ONE SHOULD EVER GET TO THE BOTTOM OF THE
;INTERRUPT STACK,A JUMP TO BASE0 WILL BE MADE.
;DO NOT RETURN TO BASIC,BUT KEEP THE INTERRUPT ON
;NOW AND THEN.

```
;THE FOLLOWING IS SOME STUFF FROM THE LISTING
;THAT SHOULD BE HELPFUL
;BASIC INTERPRETER RELLOCATABLE EXTENDED SYSTEM
;REVISED VERSION OF TSBAS WHICH IS
;COPYRIGHT (C) 1971, DATA GENERAL CORPORATION
.TITL RESUB
;DEFINE SOME USEFUL FUNCTION ENTRIES
;.ENT .FIX,.FLOT,.SUBF,.ADDF,.DIVF,.MPYF
;.ENT Z,A        ;PROVIDE ENTRIES FOR DEBUGGIN WITHOUT SYMBOLS
;DEFINE ENTRIES FOR IO SYSTEM
;IO SYSTEM MUST BE LOADED BEFORE THE BASIC SYSTEM
;.ENT C377,CMTH,INTR,INTEN,C177
;.EXTN OUTPT,RDINP,DVINT,INIT,CLOSE,ICMSK
;.EXTD .DVUS,.PTP,.PTR,.DCN, .CTU
;.ENT; FILNO     ;LOCATION OF 4 DIGIT BCD LINE NUMBER
                 ;MAY BE USED FOR A FILE NUMBER WHEN USING
                 ;USER SUPPLIED I.O. DRIVER ROUTINES

;.ENT; RUP       ;TO INCREMENT RUNTIME ACCUMULATOR USE THE
                 ;FOLLOWING CODE   LDA 2 @RUP
                 ;                 MOV 2 2 SZR
                 ;                 ISZ -23 2
                 ;THIS SHOULDE BE DONE TEN TIMES A SECOND
                 ;IF TIME COMMAND IS TO WORK
;.ENT; .INTR     ;PAGE ZERO ADDRESS OF INTR
;.EXTN STRUP     ;TO EXTERNAL START UP PROGRAM
;.EXTN BGNDP     ;BACKGROUND PROGRAM
;.ENT .TASK      ;START UP AND BACKGROUND SHOULD RETURN
                 ;WITH JMP @.TASK
;.EXTN BASEV     ;BASE PROGRAM AT BOTTOM OF STACK
                 ;INTP IS TURNED ON BEFORE CALL

;.EXTN WCC       ;TO THE START OF USER SUPPLIED
                 ;ONCE ONLY CODE AT INITIAL STARTUP
                 ;AFTER BASIC IS FIRST LOADED
;.ENT WOCR       ;RETURN FROM WCC ROUTINE
;THE FOLLOWING ARE USEFUL FOR SUBSTITUTING
;LINE PRINTER FOR TELETYPE.  BASIC TAKES CARE OF
;NORMAL TELETYPE INTERRUPT CALL TO TELO
;.ENT TELO       ;TELETYPE HANDLING ROUTINE FROM INTERUPT CHAIN
;.ENT TELO1      ;ADDRESS OF DOAS 1 TTO INSTRUCTION

;.ENT IOBT       ;POINTS TO IOB
;INITIAL STARTUP OF RESUB IS AT LOCATION 20.
;THIS WILL EXECUTE THE ONCE ONLY CODE THAT
;IS LATER DESTROYED.  WHEN CONTROL IS
;TRANSFERRED TO STRUP,THE USER STARTUP ROUTINE,LOCATION
;20 CONTAINS THE FFIRST LOCATION IN HIGH CORE NOT USED BY
;RESUB AND BASIC USER SPACE.  LOCATIONS 20,21,30 ARE
;NOT USED BY RESUB WHEN THE JMP TO STRUP IS MADE

;AFTER THE INITIAL STARTUP,
;BASIC SHOULD BE RESTARTED AT LOCATION 2
;LOCATIONS 4-7 ARE SET UP BY THE ONCE ONLY CODE SO THAT
;THERE IS HOPE THAT RESUB MAY BE LOADED UNDER DOS
;IT WILL OF COURSE WIPE OUT SECTIONS OF DOS IN CORE.



;FOLLOWING SUBROUTINES ARE USEFUL FOR STRING MANIPULATION
;.ENT .ACBY      ;INDIRECT ADDRESS IN PAGE ZERO OF SUBROUTINE
                 ;USED TO ACCESS BYTE POINTED TO BY AC1
```

```
                ;RETURN BYTE IS IN AC2

.ENT STBY       ;ADDRESS TO STORE BYTE IN AC0 USING POINTER
                ;IN AC1
```

```
;THIS PAGE IS ALSO FROM THE LISTING


;DEFINE EXTERNAL FOR SUBROUTINE TABLE
;.EXTN SBRTB, SBRTC      ;SBRTC IS ADDRESS OF TABLE
                         ;USED BY SUBROUTINES
                         ; AVAILABLE FROM THE CONSOLE
;IT MAY COINCIDE WITH SBRTB IF DESIRED OR BE A
;COMPLETELY SEPARATE TABLE.

;THIS TABLE ALONG WITH THE SUBROUTINES MUST
;BE LOADED BEFORE THE BASIC SYSTEM
;
;SUBROUTINE TABLE FORMAT:
;        .ENT SBRTB
;        .NREL
;        SBRTB:  SUBROUTINE NUMBER (0-32767.)
;                ADDRESS OF SUBROUTINE
;                VARIABLE CONTROL WORD (SEE BELOW)
;                ...
;                ...
;                -1      ;IN SUBR NUMBER POSITION
;
;FORMAT OF VARIABLE CONTROL WORD:
;        EACH EXPRESSION PRESENTED TO THE CALL STATEMENT
;(NOT COUNTING THE SUBROUTINE NUMBER) (UP TO A
;MAXIMUM OF 8 EXPRESSIONS) HAS ASSOCIATED WITH IT A
;2 BIT FIELD IN THE VARIABLE CONTROL WORD STARTING
;WITH BITS 0,1 FOR THE FIRST EXPRESSION.
;(0)     00 SIGNIFIES NO MORE VARIABLES
;(1)     01  CORRESPONDING EXPRESSION MUST BE A STRING
;(2)     10 CORRESPONDING EXPRESSION MAY BE ANY VALID EXPRESSION
;           IN WHICH ALL OF THE TERMS ARE DEFINED
;(3)     11 CORRESPONDING EXPRESSION MUST BE A SIMPLE
;           OR DIMENSIONED VARIABLE (USED WHEN THIS EXPRESSION
;           IS TO RECEIVE THE RESULTS OF THE SUBROUTINE)
;
;FOR EXAMPLE, IF A SUBROUTINE PRODUCED ONE VALUE TO BE
;STORED INTO THE FIRST VARIABLE IN THE "CALL" AND REQUIRED
;3 ADDITIONAL ARGUMENTS, IT'S CALL WOULD BE
;        CALL SUBR#,RESULT,INPUT1,INPUT2,INPUT3
;THE VARIABLE CONTROL WORD WOULD BE AS FOLLOWS
; .RDX 4
;        322207  ;FIRST VARIABLE MUST BE SIMPLE OR DIMENSIONED
;.RDX            ;RESTORE ORIGINAL RADIX
;THE BASIC WILL DETECT AS ERRORS, THE FOLLOWING CALLS TO THIS
;SUBROUTINE
;        CALL A,C+B,Y,Z,W                    FIRST EXPRESSION
;                                             MUST BE SIMPLE
;        CALL A,B,C,D,E,W                    ;TOO MANY EXPRESSIONS
;        CALL A,B,C                  ;NO ENOUGH EXPRESSIONS
;(IN THE ABOVE, A CONTAINS THE SUBROUTINE NUMBER)
```

```
;5. CALLING ASSEMBLY LANGUAGE SUBROUTINES
;ADDITIONAL NOTES
;THE CALL HAS BEEN EXPANDED SO THAT STRINGS MAY
;BE CALLED. WHEN USED AS A KEYBOARD COMMAND
;THE SUBROUTINES TABLE IS SBTBC.
;THIS MAY BE IDENTICAL WITH STBTB IF SO DESIRED.

;WHEN A SUBROUTINE IS CALLED FROM BASIC
;THE CALL IS MADE BY
; JSR TO S/R ENTRY
; ADDRESS OF FIRST PARAMETER(THIS IS IN AC3)
; ADDRESS OF 2ND
;    . . . .
; ADDRESS OF LAST PARAMETER(UP TO EIGHT)
; RETURN TO HERE FROM ASSEMBLY LANGUAGE ROUTINE

; NUMERIC DATA IS STORED IN A TWO WORD FLOATING
; POINT FORMAT WHOSE ADDRESSES ARE GIVEN AS SHOWN ABOVE
; A STRING VARIABLE IS A BIT DIFFERENT!
; INSTEAD OF HAVING A PARAMETER ADDRESS
;THERE IS A POINTER TO THE FIRST WORD OF A FOUR WORD
; TABLE WHOSE CONTENTS ARE AS FOLLOWS:

; FIRST BYTE
; DIMENSION
; FIRST SUBSCRIPT-1 OR ZERO
; SECOND SUBSCRIPT OR STRING DIMENSION

;YOU MUST BE CAREFUL WHEN SENDING STRINGS TO BASIC
;WE HAVE NOT USED THIS MUCH.  YOU MAY GET INTO
;TROUBLE IF YOU SEND CHARACTERS NOT USED IN BASIC STRINGS
;A NULL SEEMS TO BE THE TERMINATOR.  BE SURE TO SET THE
;PARITY BIT TO 1 IN THE ASCII CODES.

;THE CALL IS RESUB IS SIMILAR TO THE CALL IN
;BASIC WITH CALL BY MARTIN CAKES

;WE EXPANDED THE CALL TO BE A CONSOLE COMMAND
;WE ALSO MODIFIED IT SO THAT STRINGS
;COULD BE PASSED TO SUBROUTINES.


;THE SUBROUTINE CALL WILL GIVE ERROR MESSAGE
;37 UNDER THE FOLLOWING CONDITIONS:
;       1. IF THE APPROPRIATE SUBROUTINE TABLE IS
;                NOT LOADED.
;       2. IF THE SUBROUTINE NUMBER IS NOT IN
;                THE TABLE.
;       3. IF THE SUBROUTINE NUMBER IS IN THE
;                TABLE ;BUT THE SUBROUTINE ADDRESS
;                IS AN UNSATISFIED EXTERNAL.
;THE CALL WILL ALSO CHECK FOR THE CORRECT PARAMETERS
;AND OTHER ERROR MESSAGES CAN BE GENERATED.


; 6. INTERRUPT STACK--THE STACK MAKES IT
;POSSIBLE TO HAVE NESTED INTERRUPTS. THERE IS SPACE
;ALLOWED FOR 10(DECIMAL) STACK FRAMES.
;IT IS ALLOWED TO TURN ON THE INTERRUPT SYSTEM
;AFTER AN INTERRUPTING
;DEVICE HAS BEEN CLEARED.  IF YOU WISH TO WAIT FOR
;AN IO DEVICE TO COMPLETE IT TASK,YOU MAY
;DO A JMP @.INTR WHERE .INTR IS IN PAGE
```

```
;8. LOADING AND DUMPING PROGRAMS WRITTEN IN BASIC

;BASIC PROGRAMS MAY BE DUMPED AND LOADED
;ON SPECIAL IO DEVICES WITH USER SUPPLIED ASSEMBLY
;LANGUAGE DRIVER ROUTINES.  EXAMPLES OF PROGRAMS FOR
;PTP AND PTR ARE GIVEN IN THE SAMPLE SUBROUTINES.
;EACH CHANNEL IMPLEMENTS MUST HAVE A DEVICE CONTROL
;TABLE (DCT) WHOSE ADDRESS IS IN .ZREL. THE
;DEVICE IN USE FLAG DISPLACEMENT MUST BE DEFINED BY THE
;USER AND DECLARED AS AN ENTRY POINT.

;INPUT COMMANDS          DCT POINTER
;       TAPE             .PTR
;       CARD             .CDR

;OUTPUT COMMANDS         DCT POINTER
;       PUNCH            .PTP
;       DUMP             .CDR

;IF THE DRIVER IS NOT LOADED,MAKE SURE THAT THE
;DCT POINTER IS ZERO.IF YOU DO NOT DEFINE
;AND ENTRY POINT, THE LOADER WILL ASSIGN ITS
;ADDRESS TO 377.  (SO DO NOT USE LOCATION 377)
;THE RELOCATABLE LOADER WILL SET 377 TO ZERO
;UNLESS IT IS USED FOR SOMETHING IN .ZREL,ETC.

;9. PUNCHING ABSOLUTE VERSIONS OF RESLB:
;ONCE A SET OF SUBROUTINES HAVE BEEN DEVELOPED
;FOR USE WITH RESUB, IT IS CONVENIENT TO HAVE
;AN ABSOLUTE TAPE OF THE SUBROUTINES AND RESLB.
;LOAD THE PROGRAMS IN THE FOLLOWING ORDER
;       1. ALL USER SUPPLIED ASSEMBLY LANGUAGE PORGRAMS.
;       2.  RESUB
;       3.  DEBUG 3

;NOTE THAT DEBUG 3 IS LOADED AFTER RESLB SINCE IT WILL NOT BE
;PUNCHED.
;NOW OBTAIN A LOADER MAP FROM THE RELOCATABLE LOADER
;AND START DEBUG 3(DO NOT START RESUB)
;PUNCH OUT ALL LOCATIONS FROM 20 UP TO DEBUG.
;(IN GENERAL YOU DO NOT HAVE TO PUNCH ALL OF PAGE ZERO
;BUT IT IS NOT WORTH WHILE TRYING TO SAVE A
;FEW WORDS ON THE OBJECT TAPE.)
;SEE THE NOTE ABOUT LOCATION 377.
```

```
;8. LOADING AND DUMPING PROGRAMS WRITTEN IN BASIC

;BASIC PROGRAMS MAY BE DUMPED AND LOADED
;ON SPECIAL IO DEVICES WITH USER SUPPLIED ASSEMBLY
;LANGUAGE DRIVER ROUTINES.  EXAMPLES OF PROGRAMS FOR
;PTP AND PTR ARE GIVEN IN THE SAMPLE SUBROUTINES.
;EACH CHANNEL IMPLEMENTS MUST HAVE A DEVICE CONTROL
;TABLE (DCT) WHOSE ADDRESS IS IN .ZREL. THE
;DEVICE IN USE FLAG DISPLACEMENT MUST BE DEFINED BY THE
;USER AND DECLARED AS AN ENTRY POINT.

;INPUT COMMANDS          DCT POINTER
;       TAPE             .PTR
;       CARD             .CDR

;OUTPUT COMMANDS         DCT POINTER
;       PUNCH            .PTP
;       DUMP             .CDP

;IF THE DRIVER IS NOT LOADED,MAKE SURE THAT THE
;DCT POINTER IS ZERO.IF YOU DO NOT DEFINE
;AND ENTRY POINT, THE LOADER WILL ASSIGN ITS
;ADDRESS TO 377.  (SO DO NOT USE LOCATION 377)
;THE RELOCATABLE LOADER WILL SET 377 TO ZERO
;UNLESS IT IS USED FOR SOMETHING IN .ZREL,ETC.

;9. PUNCHING ABSOLUTE VERSIONS OF RESLB:
;ONCE A SET OF SUBROUTINES HAVE BEEN DEVELOPED
;FOR USE WITH RESUB, IT IS CONVENIENT TO HAVE
;AN ABSOLUTE TAPE OF THE SUBROUTINES AND RESLB.
;LOAD THE PROGRAMS IN THE FOLLOWING ORDER
;       1. ALL USER SUPPLIED ASSEMBLY LANGUAGE PORGRAMS.
;       2.  RESUB
;       3.  DEBUG 3

;NOTE THAT DEBUG 3 IS LOADED AFTER RESLB SINCE IT WILL NOT BE
;PUNCHED.
;NOW OBTAIN A LOADER MAP FROM THE RELOCATABLE LOADER
;AND START DEBUG 3(DO NOT START RESUB)
;PUNCH OUT ALL LOCATIONS FROM 20 UP TO DEBUG.
;(IN GENERAL YOU DO NOT HAVE TO PUNCH ALL OF PAGE ZERO
;BUT IT IS NOT WORTH WHILE TRYING TO SAVE A
;FEW WORDS ON THE OBJECT TAPE.)
;SEE THE NOTE ABOUT LOCATION 377.
```

```
;HERE ARE THE ERROR MESSAGES
;RESLB WILL CHECK FOR UNSATISFIED EXTERNALS
;IN THE SUBROUTINE TABLES. YOU GET ERROR 37 IF EITHER
;THE SUBROUTINE IS NOT LOADED OR THE TABLE IS NOT LOADED

;00      FORMAT ERROR                                    E
;01      CANNOT RECOGNIZE WORD                           E
;02      SYNTAX ERROR                                    S
;03      ILLEGAL STRING OPERATION                        S
;04      SYSTEM FAULT                                    A
;05      ILLEGAL STATEMENT NUMBER                        E
;06      TOO MANY VARIABLE NAMES                         E
;07      CAN NOT EXECUTE FROM IO DEVICE                  E
;08      DEVICE DRIVER NOT LOADED                        E
;09      DEVICE IS BUSY                                  E
;10      INCORRECT SUBSCRIPT CLOSURE                     E
;11      INCORRECT PARENTHESIS CLOSURE                   E
;12      CAN NOT EXECUTE FROM KEYBOARD                   E
;13      NO SUCH LINE NUMBER                             E,A
;14      STORAGE OVERFLOW WHILE INPUTTING PROGRM         E
;15      READ IS OUT OF DATA                             A
;16      ARITHMETIC ERROR                                S
;17      REFERENCE TO UN-ASSIGNED VARIABLE               A
;18      GOSUBS NESTED TOO DEEP                          A
;19      RETURN WITHOUT GOSUB                            A
;20      FORS NESTED TOO DEEP                            A
;21      FOR WITHOUT NEXT                                A
;22      NEXT WITHOUT FOR                                A
;23      OUT OF STORAGE WHILE ASSIGNING
;         VARIABLE SPACE                                 A
;24      ARRAY TOO LARGE FOR SYSTEM(>1224 ELEMENTS
;       OR DIMENSION >256)                              S
;25      ATTEMPT TO DIMENSION SIMPLE VARIABLE            S
;26      NAME NOT DIMENSIONABLE                          S
;27      STRING VARIABLE CAN HAVE ONLY 1 DIMENSION       S
;28      ARRAY EXCEEDS INITIAL DIMENSION                 S
;29      EXPRESSION TOO COMPLEX FOR EVALUATION           A
;30      SYNTAX ERROR IN DEFINED FUNCTION                A
;31      SUBSCRIPT EXCEEDS DIMENSION                     A
;32      UNDEFINED USER FUNCTION                         S
;33      FUNCTIONS NESTED TOO DEEP                       A
;34      ILLEGAL SUBSCRIPT (NEGATIVE OR TOO LARGE)       A
;35      STRING DIMENSION LARGER THAN STRING SIZE        S
;36      STRING TO BE PRINTED EXCEEDS PAGE SIZE          S
;37      SUBROUTINE NOT IN CORE ("CALL")                 A
;38      STRING NOT DIMENSIONED                          A
;39      SAME MATRIX CAN NOT APPEAR ON BOTH
;       SIDES (MULTIPLY OR TRANSPOSE)                   S
;40      MATRICES HAVE DIFFERENT DIMENSIONS              S
;41      MATRIX HAS A ZERO DIMENSION                     S
;42      DIMENSIONS NOT COMPATABLE (MULTIPLY)            S
;43      MATRIX IS NOT SQUARE (INVERT)                   S
;44      MATRIX HAS A 0 ON MAJOR DIAGONAL, CAN NOT
;       BE INVERTED                                     S
;45      ILLEGAL FUNCTION USAGE                          S
;46      TYPED INPUT DOESN'T MATCH "INPUT" STATEMENT
;       OR SYNTAX ERROR IN "INPUT" STATEMENT            S
;47      STRING DIMENSIONS CAN NOT BE CHANGED            S
```

```
!HERE ARE THE ERROR MESSAGES
!RESLB WILL CHECK FOR UNSATISTIFIED EXTERNALS
!IN THE SUBROUTINE TABLES. YOU GET ERROR 37 IF EITHER
!THE SUBROUTINE IS NOT LOADED OR THE TABLE IS NOT LOADED

!00     FORMAT ERROR                                    E
!01     CANNOT RECOGNIZE WORD                           E
!02     SYNTAX ERROR                                    S
!03     ILLEGAL STRING OPERATION                        S
!04     SYSTEM FAULT                                    A
!05     ILLEGAL STATEMENT NUMBER                        E
!06     TOO MANY VARIABLE NAMES                         E
!07     CAN NOT EXECUTE FROM IO DEVICE                  E
!08     DEVICE DRIVER NOT LOADED                        E
!09     DEVICE IS BUSY                                  E
!10     INCORRECT SUBSCRIPT CLOSURE                     E
!11     INCORRECT PARENTHESIS CLOSURE                   E
!12     CAN NOT EXECUTE FROM KEYBOARD                   E
!13     NO SUCH LINE NUMBER                             E,A
!14     STORAGE OVERFLOW WHILE INPUTTING PROGRM         E
!15     READ IS OUT OF DATA                             A
!16     ARITHMETIC ERROR                                S
!17     REFERENCE TO UNASSIGNED VARIABLE                A
!18     GOSUBS NESTED TOO DEEP                          A
!19     RETURN WITHOUT GOSUB                            A
!20     FORS NESTED TOO DEEP                            A
!21     FOR WITHOUT NEXT                                A
!22     NEXT WITHOUT FOR                                A
!23     OUT OF STORAGE WHILE ASSIGNING
!       VARIABLE SPACE                                  A
!24     ARRAY TOO LARGE FOR SYSTEM(>1024 ELEMENTS
!       OR DIMENSION >256)                              S
!25     ATTEMPT TO DIMENSION SIMPLE VARIABLE            S
!26     NAME NOT DIMENSIONABLE                          S
!27     STRING VARIABLE CAN HAVE ONLY 1 DIMENSIONS
!28     ARRAY EXCEEDS INITIAL DIMENSION                 S
!29     EXPRESSION TOO COMPLEX FOR EVALUATION           A
!30     SYNTAX ERROR IN DEFINED FUNCTION                A
!31     SUBSCRIPT EXCEEDS DIMENSION                     A
!32     UNDEFINED USER FUNCTION                         S
!33     FUNCTIONS NESTED TOO DEEP                       A
!34     ILLEGAL SUBSCRIPT (NEGATIVE OR TOO LARGE)A
!35     STRING DIMENSION LARGER THAN STRING SIZES
!36     STRING TO BE PRINTED EXCEEDS PAGE SIZE  S
!37     SUBROUTINE NOT IN CORE ("CALL")                 A
!38     STRING NOT DIMENSIONED                          A
!39     SAME MATRIX CAN NOT APPEAR ON BOTH
!       SIDES (MULTIPLY OR TRANSPOSE)                   S
!40     MATRICES HAVE DIFFERENT DIMENSIONS              S
!41     MATRIX HAS A ZERO DIMENSION                     S
!42     DIMENSIONS NOT COMPATABLE (MULTIPLY)S
!43     MATRIX IS NOT SQUARE (INVERT)                   S
!44     MATRIX HAS A 0 ON MAJOR DIAGONAL, CAN NOT
!       BE INVERTED                                     S
!45     ILLEGAL FUNCTION USAGE                          S
!46     TYPED INPUT DOESN'T MATCH "INPUT" STATEMENT
!       OR SYNTAX ERROR IN "INPUT" STATEMENT    S
!47     STRING DIMENSIONS CAN NOT BE CHANGED    S
```

```
;12. HOW TO MAKE AN ABSOLUTE TAPE

;ALTHOUGH RELOCATABLE TAPES ARE MUCH MORE FLEXABLE THAN
;ABSOLUTE TAPES, THERE COMES A TIME FOR EVERYBODY WHEN AN
;ABSOLUTE TAPE OF DEBUGGED PROGRAMS IS EASIER TO HANDLE.
;TO MAKE AN ABSOLUTE VERSION OF RESUB, ONE MUST
;FIRST LOAD THE USER SUPPLIED RELOCATABLE PROGRAMS, THEN
;LOAD RESUB, AND FINALLY LOAD DEBUG III.  AFTER OBTAINING
;A LOADER MAP, START DEBUG III.  DO NOT START
;RESUB.  USE DEBUG III TO PUNCH  MEMORY BETWEEN
;LOCATIONS 20 AND DEBUG-1.  PUNCH AN END BLOCK WITH
;20 DEFINED AS THE STARTING LOCATION.

;13 HOW TO AVOID QUESTIONS

;THIS IS FOR YYOU , IF YOU ARE
;TOO LAZY TO ANSWER ALL OF THE STARTUP QUESTIONS.
;WE HAVE LISTED THE QUESTIONS THAT CAN BE ANSWERED BY A "Y"
;OR AN "N".  IF YOU WISH TO SET THE ANSWER TO "N" PUT 6
;JMP .+4 INTO THE SPECIFIED LOCATION.
;USE A JMP .+5 FOR A "Y".  THIS MAY BE DONE BEFORE
;PUNCHING AN ABSOLUTE TAPE USING DEBUG III.

;LOCATION N IS GIVEN BY THE LOADER MAP.

;"SHOULD ANY WORDS AT END BE SAVED"        N+14034

;"HARDWARE MULTIPLY DIVIDE INSTALLED"      N+14067

;"DO YOU WISH TO DELETE MATRIX OPERAT.."   N+14107

;"DO YOU WISH COMMAS TO BE 14 SPACES .."   N+14311

;THE ABOVE PATCHES HAVE NOT BEEN TESTED YET.  GOOD LUCK.
.END
```