United States Department of Commerce
National Institute of Standards and Technology

NISTIR 3978

AUTOMATIC WAVEFORM ANALYSIS AND MEASUREMENT SYSTEM USER MANUAL

S.M. Chesnut
N.G. Paulter

NISTIR 3978

AUTOMATIC WAVEFORM ANALYSIS AND MEASUREMENT SYSTEM USER MANUAL

S.M. Chesnut
N.G. Paulter

Electromagnetic Fields Division
Electronics and Electrical Engineering Laboratory
National Institute of Standards and Technology
Boulder, Colorado 80303

December 1991

U.S. DEPARTMENT OF COMMERCE, Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY, John W. Lyons, Director

CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

This work was supported by the Calibration Coordination Group, project number 276, for the Army Primary Standards Laboratory at the Redstone Arsenal, Alabama.

We use trade names to specify the equipment used in this system. No endorsement by the National Institute of Standards and Technology is implied. Similar products by other manufacturers may work as well or better.

1.0 INTRODUCTION

1.1 General

The Automatic Waveform Analysis and Measurement System (AWAMS) is a system used at the National Institute of Standards and Technology (NIST) to obtain measurements of electrical pulses that have transients ranging from 10 nanoseconds (ns) to 20 picoseconds (ps) and to analyze these measurement data to obtain pulse parameters, such as transition duration (rise and fall time), pulse aberration (overshoot and undershoot), pulse amplitude (topline minus baseline), pulse duration (pulse width), etc. The AWAMS was constructed for the Calibration Coordination Group (CCG), project number 276, by NIST.

1.1 Overview of The Manual

The remainder of the User's Manual provides the information necessary to operate the AWAMS. Section 2 includes a brief review of pertinent technical information, such as the theory of deconvolution. Sections 3 and 4 describe the hardware and software components of the AWAMS and their operation. In Section 5, we give warnings and considerations on the use of the AWAMS. Section 6 contains references to related technical papers, Section 7 a glossary of terms, and Section 9 a specifications list, manufacturers users' manuals list, and example measurement procedures. Section 10 contains the software source code listings.

2.0 BACKGROUND

2.1 General

The AWAMS uses an equivalent-time sampling oscilloscope to acquire discrete-time data that will represent a periodic continuous-time waveform. Before sampling the waveform, however, the sampling oscilloscope must be readied. Readiness is provided by triggering the oscilloscope before the waveform arrives at the oscilloscope's sampling gate. Only one sample point on the waveform is taken after each trigger event. The next point along the waveform is obtained by adding a small delay between trigger and waveform arrival. The entire waveform is then mapped by successively increasing this delay for each subsequent point. To get an accurate representation of the waveform, the time interval between trigger and waveform arrival must not vary randomly. In reality, however, this time interval does vary randomly; this random variation between the time of arrival of the waveform and its trigger is called jitter. The effect of jitter is to smooth the data, as would be done by a filter. This smoothing of the data is described mathematically by the convolution of the filter with the data.

The AWAMS provides a mechanism (deconvolution) for removing the effect of jitter on the acquired data so that a more accurate representation of the input waveform is obtained. In addition, the effect of the oscilloscope's sampling aperture is also removed, as are the effects of time and voltage errors. In Section 2.2, we provide a review of sampling theory, and review the convolution and deconvolution processes in Section 2.3. We describe the time and voltage calibration procedures in Sections 2.4 and 2.5, and describe the pulse parameter calculations in Section 2.6.

2.2 Sampling

The effect of equivalent-time sampling on a continuous function of time is to discretize the time over which the function is defined. For example, if a continuous function (corresponding to some real signal) is sampled once every second, then the discretized replica of the continuous function has information only at each 1 s interval. This can be written as

$$g_m = g(m\epsilon), \quad 0 \le m \le \infty , \tag{1}$$

where m is an integer, ϵ is the time interval between samplings, $g(m\epsilon)$ is g(t) defined only at $t = m\epsilon$, g(t) is the continuous function, and g_m is the discretized replica of $g(m\epsilon)$. In eq (1), it is assumed g(t) does not exist before t = 0. Equation (1) may also be written, in a more useful form [1], as the product of a continuous function with a periodically spaced delta function $\delta(t)$:

$$g_m = g(t) \sum_{m=0}^{\infty} \delta(t - m\epsilon) . \tag{2}$$

Recall that $\delta(t-a) = 1$ when $t-a = 0$ and is 0 otherwise. It is worthwhile to examine the frequency representation of eq (2); this is done using the Fourier transform. The Fourier transform of the summation in eq (2) becomes [2]:

$$FT\left[\sum_{m=0}^{\infty} \delta(t - m\epsilon)\right] = (2\pi/\epsilon) \sum_{k=-\infty}^{\infty} \delta(t - kf_s), \tag{3}$$

where k is an integer, FT denotes a Fourier transform operation, and $f_s = 2\pi/\epsilon$ is the frequency corresponding to the sampling interval. The Fourier transform of g(t) is simply $(1/2\pi)G(f)$. By an identity of Fourier transforms, the point-by-point multiplication of two functions in one domain becomes the convolution of the Fourier transforms of the two functions in the

transform domain. Accordingly, the frequency representation of eq (2) is

$$G_k = (1/\epsilon)\, G(f) * \sum_{k=-\infty}^{\infty} \delta(f-kf_s), \qquad (4)$$

where * denotes a convolution and G_k is the discrete Fourier transform of g_m. Another property of Fourier transforms is that the Fourier transform of a real-valued waveform is Hermitian; that is, $G^*(-f) = G(f)$. Therefore, $|G^*(-f)| = |G(f)|$. (The superscript * denotes a complex conjugate, and the vertical bars indicate absolute values.) Let f_N be the band-limiting frequency of G(f); that is, f_N is the highest frequency having information on g(t). Because G(f) is Hermitian, |G(f)| is symmetric about f = 0 and has information from f_{-N} to f_N. Examination of eq (4) shows that f_S must be greater than $2f_N$ to avoid overlap between replicas of G(f) that are centered at adjacent kf_S values. This, then, is the sampling criterion: the sampling frequency must be at least twice the highest frequency that contains information on the input waveform. Accordingly, the waveform must not have faster transients than the sampling period.

2.3 Deconvolution

This section is taken from Ref. [3]. Data acquired from the measurement of a given signal are affected by the necessary intervention of the measuring device (such as measuring an electrical pulse with an oscilloscope). These data represent the signal as viewed by the measurement instrument and, therefore, can be described by the convolution of the instrument's impulse response with the signal. Consequently, it is necessary to remove the effects of the instrument on the data to get a more accurate representation of the signal; this is done by deconvolution. Discrete convolution is described by

$$f_\tau = \sum_{m=0}^{N-1} g_m h_{\tau-m} \,, \tag{5}$$

where, for illustration, g_m is a measurable characteristic of an event under investigation, h_m is the impulse response of the measurement system, f_τ is the acquired signal, m is the time index, N is the number of points in the record, and τ is the delay. The frequency domain equivalent of eq (5) is

$$F_k = G_k H_k \,, \quad 0 \le k \le N-1 \,, \tag{6}$$

where k is the frequency index and the functions F_k, G_k, and H_k are the discrete Fourier transforms of the time functions given in eq (5). Typically, deconvolution is done with the discrete Fourier transformation (DFT) of the data and instrument response and, therefore, assumes periodicity of the data. Consequently, if the waveform is step-like (a step-like waveform is a waveform that has zero or nearly zero slope at either end of the record, and the nominal values at the ends of the record are not the same) the abrupt end of the record will cause oscillations in the record of the deconvolved data. This phenomenon is expected because the abrupt transition is artificial. Therefore, in order to perform deconvolution using step-like waveforms, the record's truncation discontinuities must be dealt with properly. The procedure used at NIST to minimize the record truncation discontinuity in step-like waveforms is the Nahman-Gans record-extension technique [4]. Other techniques that may minimize errors and decrease computation time are presently under investigation. In the frequency domain, the deconvolution becomes a division of the spectra,

$$G_k = F_k / H_k, \tag{7}$$

and g_m is recovered by doing an inverse Fourier transform of G_k.

Equation (5) represents a linear convolution process whereas eq (6) represents a cyclic convolution process. The two processes provide the same result as long as time-aliasing in the cyclic process is prevented [1]. For the waveforms used here, eq (5) and (6) will give equivalent results.

If the impulse response is not known exactly, only an approximate solution or best guess can be found to the deconvolution. The deconvolution is called "blind" when G_k and H_k of (6) are not known exactly. Furthermore, more than one possible solution may exist for the blind deconvolution, and the best solution must be selected. For example, when an electrical waveform is measured with an oscilloscope, the oscilloscope's impulse response is usually not known but is approximated. Consequently, either the data may be accepted as a true representation of the input signal or deconvolution attempted with an approximate impulse response and the dubious results accepted.

The ability to solve the blind deconvolution problem and obtain an accurate approximation to the signal is very important, especially when the instrument has a large effect on the signal. (Actually, blind deconvolution is not solved; only the best-guess solution is obtained.) The best-guess solution to the blind deconvolution problem is obtained by iterative techniques. The iterations continue until a change in a predetermined waveform attribute occurs. This attribute change is used as an indicator of the stability of the solution and is called the stopping criterion. The change in the waveform attribute is caused by varying a parameter used in the iterative deconvolution. The technique presently used by NIST to solve the blind deconvolution is a matched-filter technique using a variable parameter we call Γ [5]. The filter is a low-pass filter, and Γ is related to the roll-off frequency. The filtering is applied to frequency domain deconvolution to suppress the noise that results by performing

deconvolution for frequencies where there is no appreciable spectral content in H_k. The stopping criterion presently in use is the minimum in the power of the imaginary part of the reconstructed record. This power is given by

$$P_I = \sum_{m=0}^{N-1} [I(g_m)]^2. \qquad (8)$$

Other filtering techniques and stopping criteria that may provide better results and shorten processing time are being investigated.

2.4 Time Calibration

The time, or horizontal, axis calibration is performed by putting a single-frequency sinusoidal waveform into the oscilloscope. The zero crossings of the sine wave are used to calibrate the time axis. The zero crossings are used because they provide the most robust sine-wave parameter in terms of resistance to noise, amplitude fluctuations, and jitter. Once the locations of the zero crossings are determined, the time axis can be calibrated. For example, consider using a 10-GHz sine wave to calibrate a 1-ns window that is defined over 1000 points. The zero crossings should be 50 points, or 0.05 ns, apart. If they are not for any region between a pair of adjacent zero-crossings, the time values are corrected in that region. Once the time axis is corrected, the data are linearly interpolated so that their new values coincide with their new time.

2.5 Voltage Calibration

The voltage calibration routine is a steady-state technique. A known dc voltage is applied across the terminals (center and ground) of the sampling head's input port, and the voltage measured by the oscilloscope is recorded. After performing multiple measurments of this type, a relationship between the actual and measured voltages is obtained. This relationship can be imagined as a plot of the actual versus measured voltages. The slope of this plot is used to correct the voltage values of the acquired data. Consider an example where the measured voltage equals 0.9 times the actual voltage. The new data will be $V_c = V/0.9$, where V_c is the corrected value and V is the acquired value.

2.6 Pulse Parameter Calculation

The purpose of measuring a pulse waveform is to calculate the values of the parameters that describe the waveform features. The parameters calculated by the pulse parameters program are: pulse amplitude, pulse transition duration, overshoot, and undershoot. We use a histogram-based algorithm for pulse parameter determination. The histogram represents a distribution function of the y-axis, usually voltage, values. The pulse parameters are defined in the ANSI/IEEE (American National Standards Institute/Institute of Electrical and Electronics Engineers) Std 194-1977 [6], and the algorithms we use for calculation of the pulse parameters are based on the IEEE Std 181-1977 [7].

3.0 HARDWARE SYSTEM DESCRIPTION

3.1 General

The AWAMS hardware system consists of an digitizing oscilloscope, a computer controller, and computer peripheral devices that are linked by an IEEE 488 bus interface. A block diagram of the hardware system is shown in Fig. 1. Descriptions of the hardware used in the AWAMS are found in the manufactuers' users' manuals. A list of these documents is found in Appendix A; refer to these manuals for specifications, operation procedures and safety information. References to these manuals will be indicated by [MUD {A}], where A is the document number.

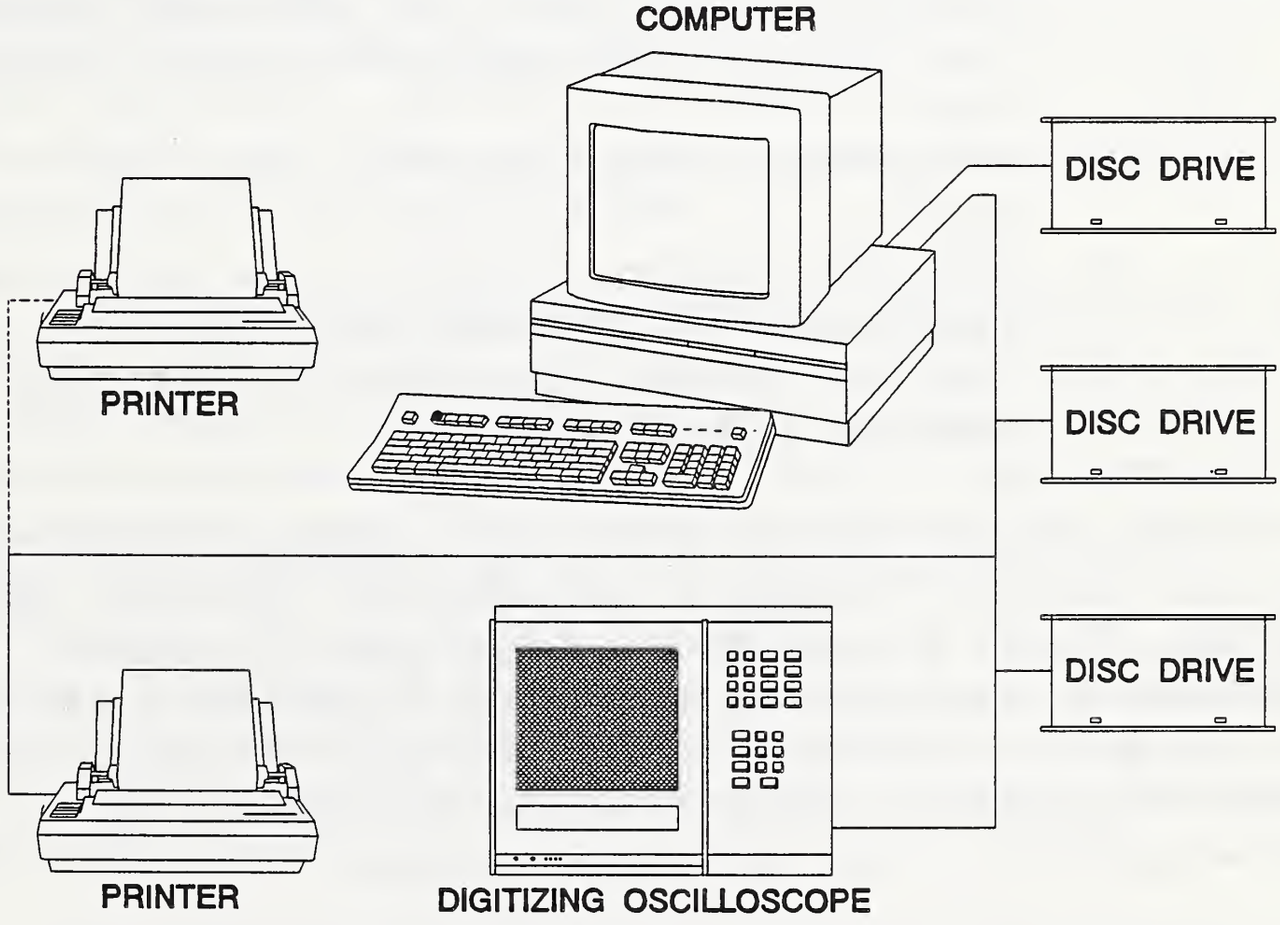

Figure 1. Diagram of AWAMS hardware system.

4.0 SOFTWARE SYSTEM DESCRIPTION

4.1 General

The following programs are included in the AWAMS package:

1. ACQUIRE: a data acquisition routine that allows for acquisition of calibration data and waveform data.
2. FIXACQ: a utility to alter the waveform according to the calibration data.
3. DECON_NIST: a deconvolution routine.
4. GAUSS: a Gaussian waveform generator that creates a waveform to approximate the distribution function of the jitter of the measurement system. This function may be deconvolved from the waveform data.
5. PULS_PARAMS: a pulse parameters calculation program.
6. MATH_OPS: a vector utilities program that performs waveform mathematical operations. And,
7. text_out: a program that prints the ASCII file generated by PULS_PARAMS.

These programs must be loaded into the computer before they are run. The BASIC language is case sensitive; therefore, type the names exactly as shown above when you load the programs. A more detailed description of each program is provided in Sections 4.2 through 4.6, including menu descriptions or flow charts and written descriptions. Program details can be found in the following text or in the source code listings.

A version of GRAPH_DATA, the graphics support program developed at NIST is provided. No documentation for GRAPH_DATA is included in this manual.

Error checking is included in the AWAMS software. If you hear a beep or series of beeps after running a program, pay attention to the message displayed on the screen. The beeps are alerting you to the presence of an error message. This message will let you know what went wrong.

4.2 Data Acquisition -- (ACQUIRE)

There are four measurements required for testing a pulse: the device under test (DUT) measurement, the voltage calibration measurement, the time calibration measurement, and a jitter measurement. The jitter measurement is an estimate of the time jitter in the DUT and measurement system [8] and is a single point measurement, described in the Appendix (A.4.4). With the exception of the jitter measurement, all measurements require the acquisition of digitized waveforms.

The acquisition program acquires the DUT waveform data, the voltage calibration data, and the time calibration data. This program is menu driven. Menu selections are made using simple on-screen graphics and the "softkeys." The main menu (softkeys) allows for the selection of any of the following: the waveform acquisition menu, the voltage calibration menu, the time calibration menu, and program information (see Fig. 2). Table 1 gives a brief description of the softkey options. No attempt was made to fully automate the front panel operation of the oscilloscope; therefore, many oscilloscope operations can only be performed manually.

The first time you enter any of the menus the acquisition program's user-definable parameters are preset to default values. When you change any parameter value, the system will retain the new values, as long as the program is in active memory (RAM), even if you move from menu to menu.

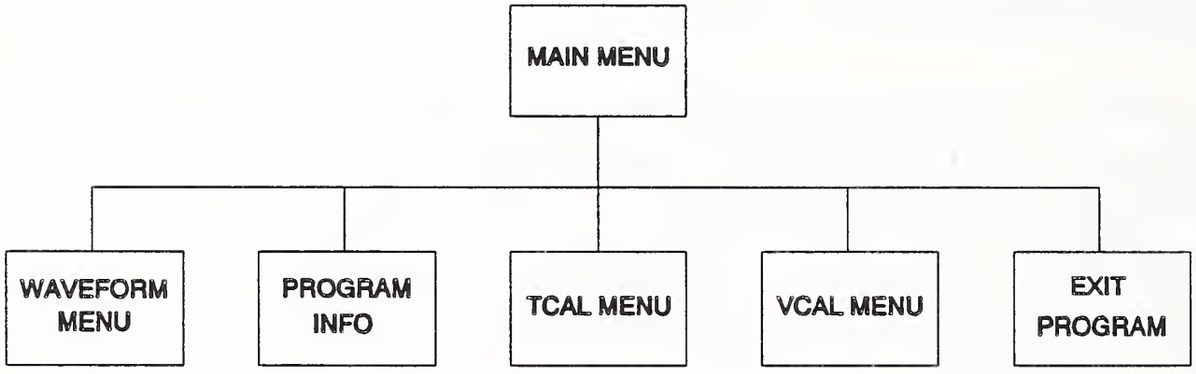

Figure 2. Softkey menu for the data acquisition program ACQUIRE.

There are oscilloscope settings that are not retained in memory. Any setup parameter not available in any menu will not be remembered when you change measurement menus. For example, if you use the oscilloscope's internal step generator as the input signal to a device, as in time-domain reflectometry (TDR), the step generator is turned off when you change to any other menu. The oscilloscope is reset every time you move from one menu to another to prevent the oscilloscope from being in an unknown state. However, to use the step generator as the input signal, use the manual setup feature of the acquisition program, which we describe later in this document, to turn on the step generator and view the waveform. Then use the "save setup" feature of the oscilloscope to retain this configuration [MUD {1}]. Acquire the

Table 1. Softkey options for the ACQUIRE program.

CHOICE	FUNCTION
Main Menu	Returns the program from current menu.
Exit Program	Quits the Acquire program.
Acquire Data	Causes the oscilloscope to take the data. The oscilloscope takes control of the bus during acquisition, but the "ABORT" key can be used to interrupt the acquisition.
Change Value	Allows for changes of the on-screen parameters. See the Tables on screen menu choices.
Manual Setup	Waveform: Allows for front panel operation of the oscilloscope. May change any parameter values and the new value will be maintained when the program is continued. TCAL/VCAL: Allows for front panel operation of the oscilloscope. The values of menu items, such as points, averages, etc., cannot be changed using this option. Other items, such as the trigger slope, that are not included in the ACQUIRE program's menu selection will change using this feature. The manual setup key is most useful for determining calibration of waveform details like establishing the length of the delay line in the TCAL trigger circuit that is used to correctly place the signal in time.

data normally and then acquire the time and voltage calibration data. When you want to acquire another set of data using the step generator, return to the waveform menu, use the manual setup key, recall the saved setup and then continue with the measurement. By using the save and recall features, the oscilloscope's parameters will be restored to the same parameters used in the previous measurement.

The parameters, in every menu, have error checking. You change some of the parameters in their own menus. Other

parameters are changed by toggling. Each of these provides error checking by limiting your alternatives to those values that the oscilloscope can provide. However, there are some parameters that are checked only for the value of the numeric input that make sense for the parameter. It is possible to input a value that the oscilloscope cannot accommodate; then the oscilloscope will default to the closest value it can provide. This is the only time there could be an inconsistency between the screen display and the acquired data. An error message will appear for a short time on the oscilloscope screen. Familiarity with the operation of the oscilloscope will help to prevent these errors.

In addition to error checking of the data, this program checks for a response from the oscilloscope, an external trigger, and the internal step generator. These checks prevent the attempted acquisition of waveform data from an oscilloscope that is off or from an oscilloscope that is on, but does not have a signal input into one of the test channels. The check for the existence of a waveform does not verify the sampling channel selection.

4.2.1 Waveform menu

First we describe the waveform menu (see Fig. 3). The parameters available in this menu are:

1. the channel for acquisition;
2. volts per division or vertical sensitivity;
3. offset or vertical position control;
4. attenuation, this facilitates the use of attenuator probes;
5. time per division or horizontal sensitivity;
6. delay or horizontal position control;
7. delay reference, either center or left side of the screen;
8. number of points to acquire in the waveform;

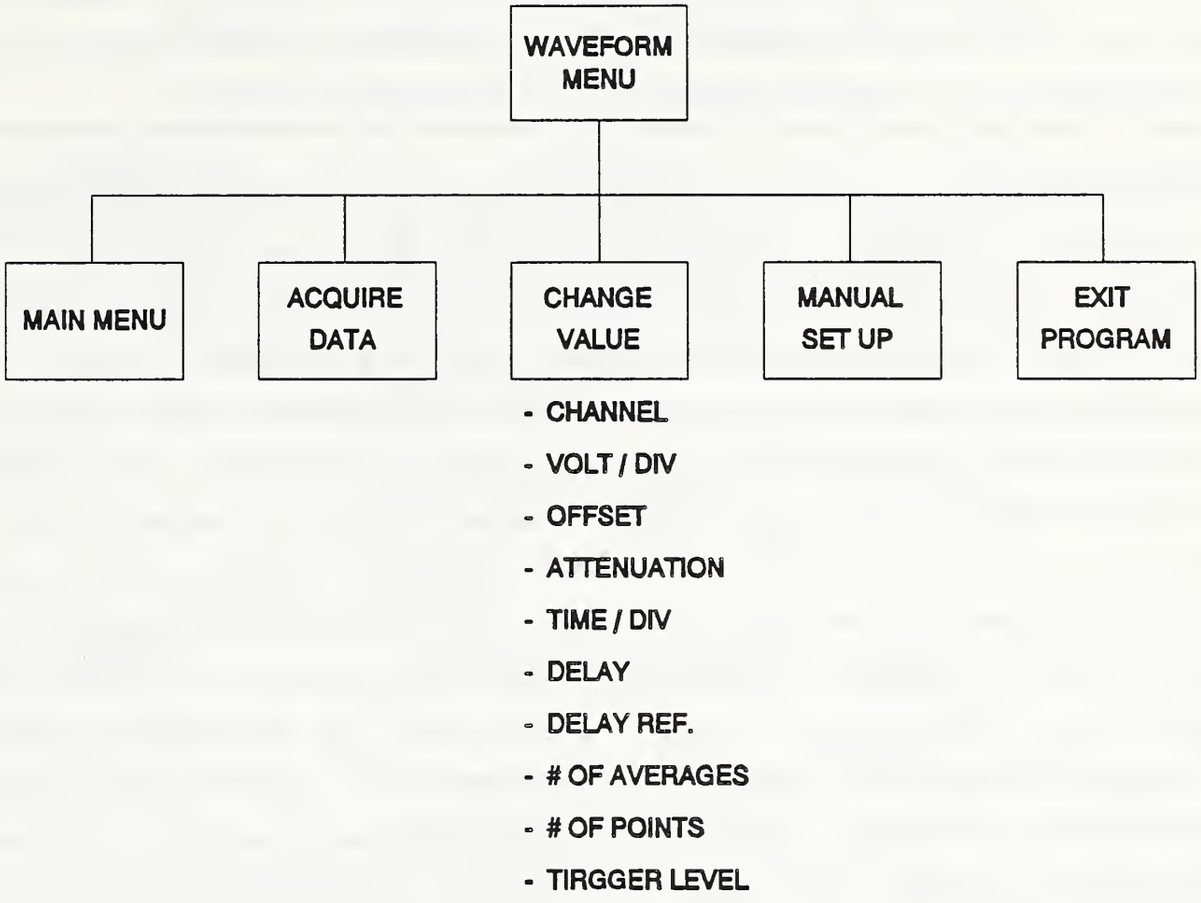

Figure 3. Waveform menu of the acquisition program.

9. number of averages used to re-create the waveform; and
10. trigger level.

The oscilloscope user manuals can provide more information on about these parameters, see [MUD {1}] and [MUD{2}].

You may select any one of the above parameters with the highlight box. The highlight box can be moved using the arrow keys, space bar, or knob. Table 2 gives a brief summary of the effect of changing each parameter and whether the values are changed from a menu, input by the operator, or toggled.

You can change the value of the selected parameter by using the softkey (at the bottom of the screen) marked "Change Value." Depending upon the parameter selected when you press the "Change

Value" key, you will be prompted to input a value, to make a selection from another menu, or the "Change Value" key will cause the parameter to toggle between two possible values.

You will choose most of the parameter values, such as: volts per division, offset, and trigger level based on the DUT waveform. For example, the offset is chosen to center the waveform on the oscilloscope screen and the trigger level is chosen for the most nearly jitter-free waveform. You will select other parameter values for more subjective reasons. For example, increasing the number of averages taken during a measurement decreases the noise in the result. Another example, the number of points selected is based on the resolution you require. The trade-off for a larger number of averages and/or a larger number of points is acquisition time. Generally, we recommend selecting the highest available number of averages and points. The only instance when a large acquisition time is detrimental to the measurement is when the waveform drifts. However, if the waveform drift is significant, a high quality measurement is not possible under any circumstance.

Table 2. Waveform screen menu selections.

SELECTION	FUNCTION	HOW TO CHANGE
Channel Number	Selects a channel for acquisition.	Keyboard input. Range: 1 to 4.
Volts per Division	Changes oscilloscope display vertical resolution.	Keyboard input Range: 1 to 80 mV
Offset	Changes the vertical placement of the waveform trace--used to correct a dc offset.	Keyboard input. Range: ± 500 mV
Attenuation	See description and warnings, Secs. 5-3 Ref. [6].	Keyboard input. Range: 1 to 1000
Time per Division	Changes oscilloscope display horizontal resolution.	Keyboard input. Range: 10 ps to 1 s
Delay	Controls the time position of the waveform.	Keyboard input. Options are time-window dependent.
Delay Ref	Selects position (center or left side of screen).	Toggle.
Number of Averages	Changes number of averages used to acquire waveform.	Menu. 1 to 2048, in steps of powers of 2.
Trigger Level	Changes oscilloscope trigger level.	Keyboard input. Range: ± 1 V

In addition to the screen menu, the waveform menu also includes the following softkey options:

1. exit to "Main Menu,"
2. "Exit Program,"
3. "Acquire Data,"
4. "Manual Setup," and
5. "Change Value."

We described the "Change Value" key above. The "Exit Program" and "Main Menu" keys are self-explanatory. The "Exit Program" key must be pressed to quit the program.

The "Manual Set" option allows you to use the front panel input keys to adjust the oscilloscope. After modifying the measurement setup, press the "Continue" key to resume the program. The "Continue" key is either a softkey that appears on the screen, or a hard key on the keyboard. When the "Continue" key is pressed, all of the screen options are updated to the values chosen while in manual mode and the oscilloscope is returned to the remote mode. Be aware that changes made to parameters that are not included in the on-screen menu will need to be reset if you leave and then return to the waveform menu.

Selecting the "Acquire Data" key will reset the oscilloscope to the waveform parameters displayed on the screen, acquire the data, and send the data to the computer for storage. The data are stored in a format compatible with GRAPH_DATA. The oscilloscope will assume control of the IEEE-488 bus during the acquisition operation.

There is an "Abort" softkey available during acquisition. This option may be selected if you need computer control or if a parameter was incorrectly selected. When you select the "Abort" key, the measurement stops, the oscilloscope is reset, control of the bus is returned to the computer, and the program returns to the waveform menu.

4.2.2 Time-calibration menu

The second menu we describe is the time-calibration or "Tcal" menu (see Fig. 4) that is used to acquire the time-base calibration data. This menu is very similar to the waveform-acquisition menu, but it contains only those parameters that are

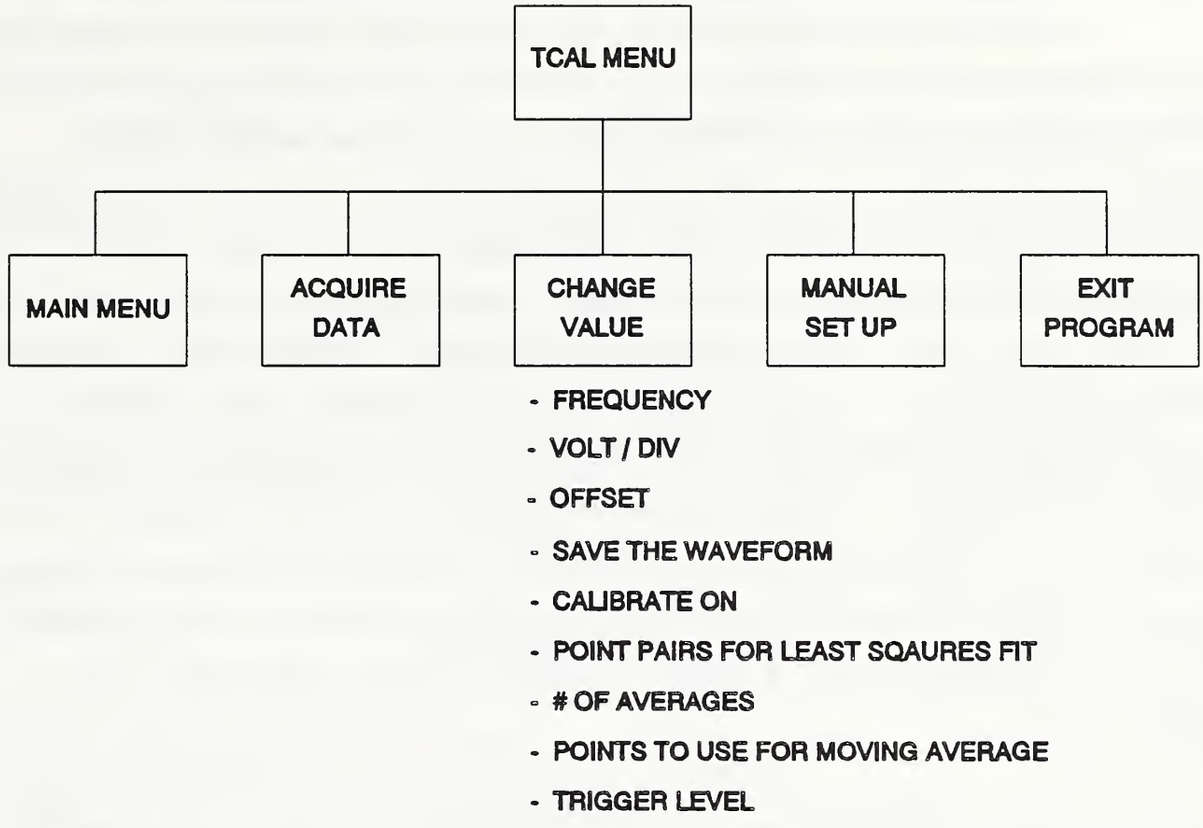

Figure 4. TCAL menu of the acquisition program.

pertinent to time-base calibration and allows changes to those parameters that will not compromise the consistency of the data set. The time calibration screen choices are:

1. calibration frequency (the frequency of your time standard);
2. number of averages for acquisition;
3. volts per division;
4. offset;
5. the number of points for the sliding average noise filter;
6. the number of points for the linear least squares fit (for finding the zero crossings of the calibration waveform);
7. the slope (positive or negative) to calibrate on;

8. whether or not to save the waveform from which the time-calibration data are derived (this is generally a sine-wave); and
9. the trigger level for the time standard.

As with the waveform menu, you can select the above parameters by using the arrow keys, space bar or knob and the highlight box. When you press the "Change Value" key, you will be prompted to input a value, to make a selection from another menu, or the "Change Value" key will cause the parameter to toggle between two choices. Again, the behavior of the "Change Value" key depends on the selected parameter. Table 3 gives a brief summary of the effect of changing each parameter and whether the values are changed from a menu, input by the operator, or toggled.

The time calibration menu also includes the following softkey choices:

1. exit to "Main Menu,"
2. "Exit Program,"
3. "Acquire Data,"
4. "Manual Set," and
5. "Change Value."

With the exception of the "Manual Set" key, these softkeys work as described in the waveform menu section. The "Manual Set" key in the time calibration menu is available to facilitate the setup of the time calibration waveform in the epoch. With this feature, you can reposition the calibration waveform with external delay line(s), usually inserted in the trigger circuit, and quickly see the effect of the added delay line. If you change the parameter values while in the manual mode, none of the changed values are read into the acquisition program. All changed settings are ignored when you return to the program from "Manual Set." This feature provides a fast and convenient way for you to determine the optimal physical setup and built-in

Table 3. TCAL screen menu selections.

SELECTION	FUNCTION	HOW TO CHANGE
Calibration Frequency	Establishes the frequency for time calibration.	Keyboard input. Depends on time calibration standard used.
Number of Averages	Changes number of averages used to acquire waveform.	Menu. 1 to 2048 in steps of powers of 2.
Volts per Division	Changes oscilloscope display vertical resolution.	Keyboard input. Range: 1 to 80 mV.
Offset	Changes the vertical placement of the waveform trace.	Keyboard input. Range: ± 500 mV
Number of Points for Sliding Average Filter	Used to filter noise and find voltage crossings.	Keyboard input.
Number of Points for Least Squares Fit	Used to find the time of the zero crossings.	Keyboard input.
Slope	Used to find the time of the zero crossings.	Toggle: +/-
Save the Time Calibration Waveform	Save time-calibration waveform data.	Toggle: yes/no
Trigger Level	Changes oscilloscope trigger level.	Keyboard input. Range: ± 1 V.

protection against acquiring inconsistent data. You must acquire the time calibration data using the same time/division and delay settings as the DUT data. If not, the epoch for the DUT will not be the same as the calibrated epoch and the data set will be useless. For more details on establishing the time calibration setup, see Sections A.4.2 and A.4.3 of this manual.

4.2.3 Voltage-calibration menu

The last menu available in the acquisition program is the "Vcal" or voltage-calibration menu (see Fig. 5). This is the simplest menu in the program with only four screen options and five softkey options.

The four screen options are:

1. number of voltage intervals to measure;
2. the number of averages wanted for the acquisition;
3. the starting calibration voltage value; and
4. the step size of the calibration voltage increments.

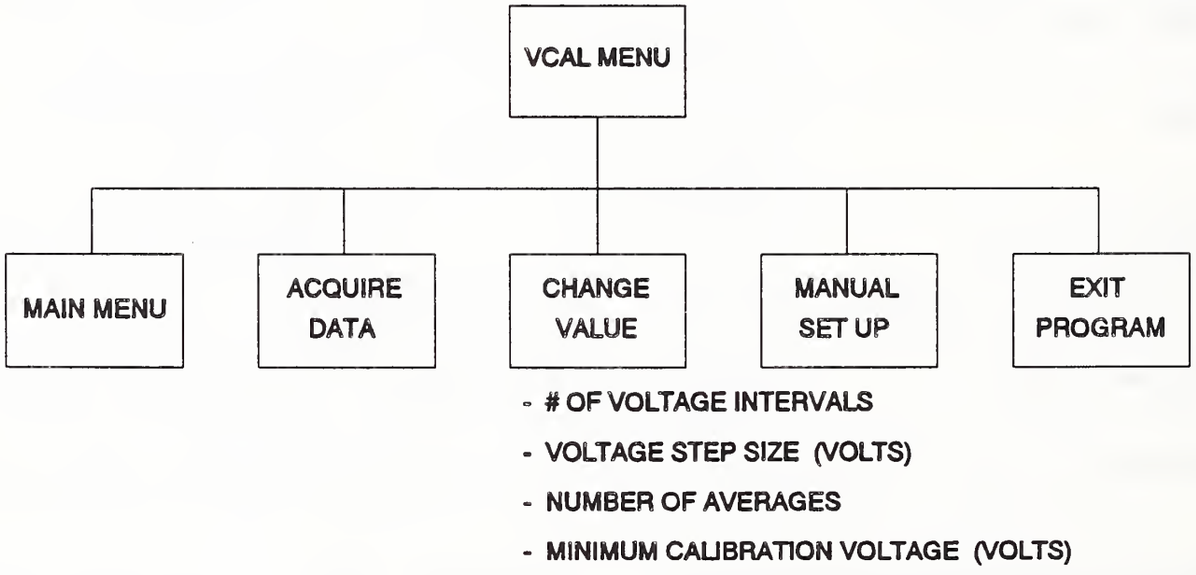

Figure 5. VCAL menu of the acquisition program.

Table 4. VCAL screen menu selections.

SELECTION	FUNCTION	HOW TO CHANGE
Number of Voltage Levels	To set the voltage range for calibration, used with Step size.	Keyboard input. Range: >0
Averages	See waveform menu.	Menu.
Starting Calibration Voltage	To set starting value that is used to calibrate data.	Keyboard. Values depend on voltage standard.
Step Size	To set the voltage increment for calibration.	Keyboard. Values depend on voltage standard.

As with the waveform and time-calibration menus, you may select these parameters by using the highlight box. Table 4 gives a brief summary of the effect of changing each parameter and whether the values are changed from a menu, input by the operator, or toggled.

The five softkey choices are:

1. exit to "Main Menu,"
2. "Exit Program,"
3. "Acquire Data,"
4. "Manual Set," and
5. "Change Value."

These keys work as described in the time calibration acquisition section. Again for data consistency, the Manual key does not result in permanent changes. The "Manual Set" key in this menu provides a convenient method for determining the

voltage calibration interval, step size, and minimum voltage. See Sections A.4.2 and A.4.3 of this manual for more details.

4.2.4 Operator information

We provide this feature of the acquisition program to give new system users some idea of the programs function and to highlight a few of the more important considerations of data acquisition using the oscilloscope. This is a limited "Help" function. You may alter these messages to fit your needs by altering the source code. We do not intend this attribute as a replacement for reading this manual or the manuals for the system hardware. Any information contained in the original message when you select the operator information softkey is also in this manual.

4.3 Oscilloscope Calibration -- (FIXACQ)

FIXACQ is the program for applying the voltage and time calibration data to the DUT waveform. You may elect to calibrate the voltage scale, the time scale, or both (see Fig. 6). If you decide to apply both voltage and time correction to the DUT data, the voltage calibration will always be done before the time calibration.

Once the program starts, you will be prompted to input the appropriate file names. Error checking is provided to insure that you do not attempt to use an empty file.

A confusing situation occurs when the calibration epoch is less than the measured data epoch. This creates the situation in which the measured data extend beyond the calibration data and the question arises as to what to do with the "extra" measured data. Should this occur, you will have seven options. These

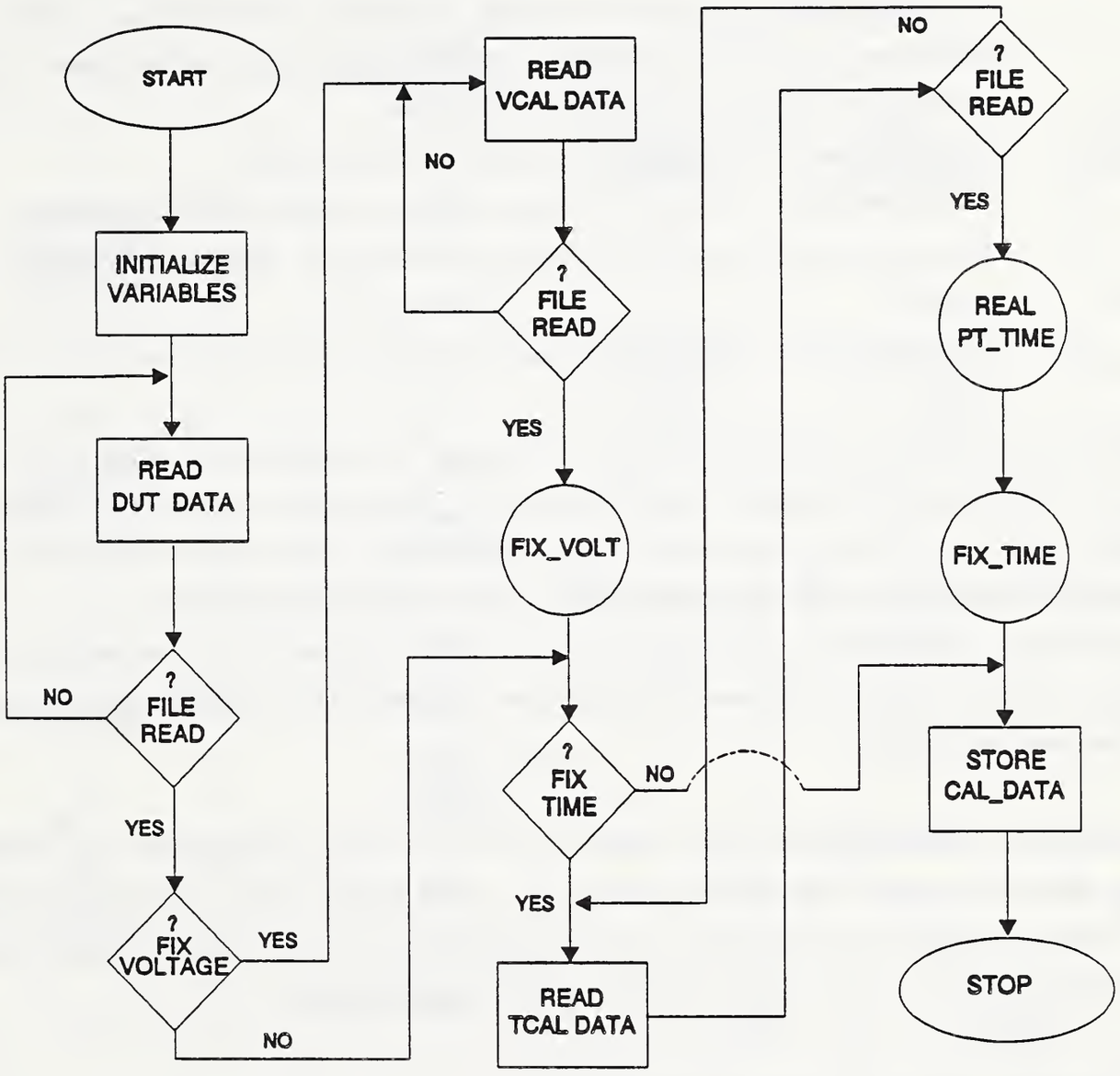

Figure 6. FIXACQ flow chart.

options are listed below.

0. Abort the calibration; no new data will be saved.
1. Abort only the time base calibration. This will save the voltage calibrated data.
2. Calibrate and output fewer data points. This is not recommended since the fast Fourier transform (FFT) routine used in deconvolution requires waveforms that are 2^n points in length.
3. Extrapolate using the last data point.
4. Extrapolate using a value input from the keyboard.
5. Extrapolate using the mean value of the last 5% of the data.
6. Extrapolate using the mean slope of the last 5% of the data.

In this program, extrapolation is done by replacing the uncorrected points with the value for the given option. You will choose option 2 when you want to calibrate the DUT waveform and have no intention of deconvolving the calibrated data. Generally, we select option 6 as a "best guess" for what the data would actually be, unless there are compelling reasons for one of the other choices.

After calibrating the data, you will be prompted to input a file name for the corrected data -- the data will then be saved and the program terminated. The data are saved in binary data format and are also consistent with GRAPH_DATA.

4.4 Deconvolution -- (DECON_NIST)

You will use this deconvolution program for both the deconvolution of the jitter impulse response and the system impulse response. We provide the system-impulse-response data as part of the AWAMS. These data are complex (incompatible with GRAPH_DATA); therefore, you will need to respond accordingly when the deconvolution program asks for the data type. The flow chart

(Fig. 7) shows the outline of the steps in this routine. Further

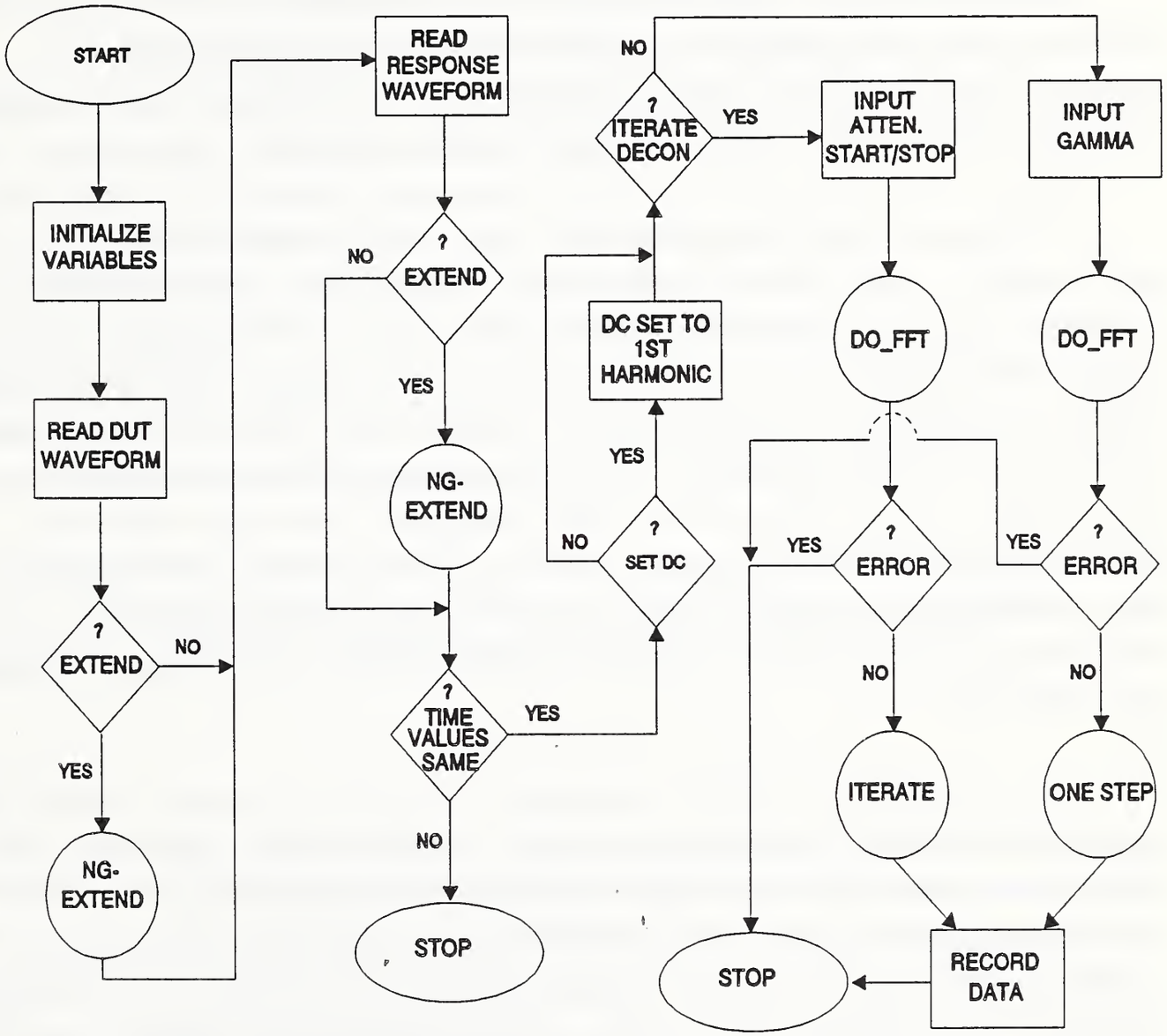

Figure 7. DECON_NIST flow chart.

details about the algorithms for waveform extension, iterative or single-pass deconvolution, and FFT preparation follow. See Section 2 of this manual and Refs. [1] through [5] for more detail on the theory of deconvolution.

Theoretically, deconvolution is a straightforward operation. However, in the presence of noise and other uncertainties, deconvolution becomes complicated [5]. Successful use of this program for deconvolution requires some insight.

First, you will be asked if you want to extend the waveform(s). If the DUT waveform is step-like, Nahman-Gans extension [4] (or another method for coping with record discontinuities [3]) is necessary for the deconvolution to work. Also, the number of points and the epoch for the DUT and response waveforms must be identical for the deconvolution to work. Therefore, if you extend the DUT waveform, you will have to extend the response waveform also. If you know a waveform has been stored in extended form, as with the system response waveform data provided as part of the AWAMS, no further extension is required.

If you extend the DUT waveform, you will also be asked what type of waveform is being extended, step-like or impulse-like. This is because the extension method is different for each of these waveform types. The extension of step-like waveform is explained in the preceding paragraph and in Section 2.3 of this manual. An impulse-like waveform is extended by adding zeros to the waveform up to the desired number of points of the extension. Therefore, the program prompts you for the appropriate input as needed, but you will need to know the right answer(s). If you request a waveform extension when one is not necessary, the program may end because of an inconsistent number of data points, or there may be no adverse effect if both the DUT and the response waveforms are unnecessarily extended.

The deconvolution program checks for consistency in the epoch and number of points acquired. The program will not perform the deconvolution and will end normally if these conditions are not met and you will have to re-run the program with a consistent data set.

The second aspect of this deconvolution program that requires explanation and decision is the choice between iterative and single-pass deconvolution. Select the single pass option *only* when you already know the optimal value of Γ. (Γ is the variable parameter that we use for selecting the optimal solution.) When the optimal value of Γ is not known, select the iterative option. The iterative option causes the program to search for the optimal solution by varying the value of Γ until our stopping criterion is met. As mentioned in Section 2, our stopping criterion is a minimum in P_i, see eq (8).

When you select the iterative option, you will be asked to input the starting and stopping values for the attenuation. The optimizing parameter, Γ, is defined by

$$\Gamma = 10^{A}, \tag{9}$$

where A is the attenuation. Do not select too large a range between the starting and stopping attenuation values. Although you may be more assured of finding the correct stopping point, iterative deconvolution takes a long time, and a larger range means a longer calculation time. It is all too easy to select a range for the attenuation that will not allow the program to find a stopping point. This happens when the range selected is too far from the correct value. Should this happen, reset the computer and run the program using a different range for the attenuation values. We are investigating other methods for finding the optimal stopping point; these may result in a more robust computation, but until then, you must develop a feeling

for what will and will not work. Typically, if the two waveforms for deconvolution are similar, the optimal value of Γ will be small (10^{-3} or smaller). On the other hand, if the waveforms are dissimilar, as is most often true, the optimal value of Γ can be quite large (10^{23} or larger). Select the starting and stopping attenuation values based on the waveforms that are being deconvolved. For instance, when deconvolving the oscilloscope system response waveform from a measured waveform, we most often choose a starting attenuation of 15 and a stopping attenuation of 20. This range is based on experience with this deconvolution algorithm.

The third aspect of DECON_NIST we describe are the routines required for performing the Fourier transforms. Unlike the extension and iteration described above, the FFT routines are transparent to the user. The first required routine is called Do_fft. Do_fft sets up the call to another routine, Fft_fix; interprets any errors detected by Fft_fix; and sets an error flag if an error has been detected. When this flag is set, the program terminates. The Fft_fix routine checks the data for situations that will generate an error in the FFT; if no errors are detected, it splits the data into its real and imaginary parts; and calls the FFT subroutine. When an error occurs, the error flag is assigned a number based on the cause of the error. This value is decoded in the Do_fft routine resulting in an on-screen error message and program termination.

Finally, we describe the options for data storage in this program. You will be given the choice to store an intermediate step and you will have the option to store several final results. All data are stored in binary data format and the stored files are compatible with GRAPH_DATA. The first result you may choose to keep is the Γ-versus-P_i curve used to determine the optimal value of Γ. This is an intermediate result that may be useful for two reasons. First, if you chose to re-run this program on

the same data set, you can use the single-pass option since you will know the optimal Γ. And second, you may want to save this information for comparison with other similar data sets or as part of your records for each device you test. After the deconvolution process is done, you will have the option of saving the spectrum magnitude, the real time-domain result, and/or the imaginary time domain result. Usually, the only result you will want or need is the real time-domain result. You may save the spectrum magnitude for any waveform type, but it is most useful for determining the frequency content of pulses used for electromagnetic interference (EMI) testing, usually an impulse-like signal. The imaginary result is made available in case you want to check the imaginary part to assure yourself that it is small; ideally the imaginary part of a real-valued waveform is 0. If you elected to extend the DUT waveform, you will also be given the option of saving the resultant waveforms in the extended or half-length forms. If you are saving a system impulse-response waveform that will later be used for another deconvolution, save the extended versions. If you are not going to do any subsequent deconvolution(s), save the unextended form because the extended version is not useful. This program will end automatically when the data have been stored.

4.5 Pulse Parameter Calculation -- (PULS_PARAMS)

Load and run PULS_PARAMS (Fig. 8) to calculate various pulse parameters. You may use this program on step-like, square-like, or impulse-like waveforms. To make the program more general, the units, such as volts and seconds, or amperes and milliseconds, are user inputs; this allows the program to operate on different kinds of data. This program calculates

1. peak to peak and 0 percent to 100 percent data values;
2. pulse amplitudes;
3. 10 percent to 90 percent and 20 percent to 80 percent pulse transition durations;

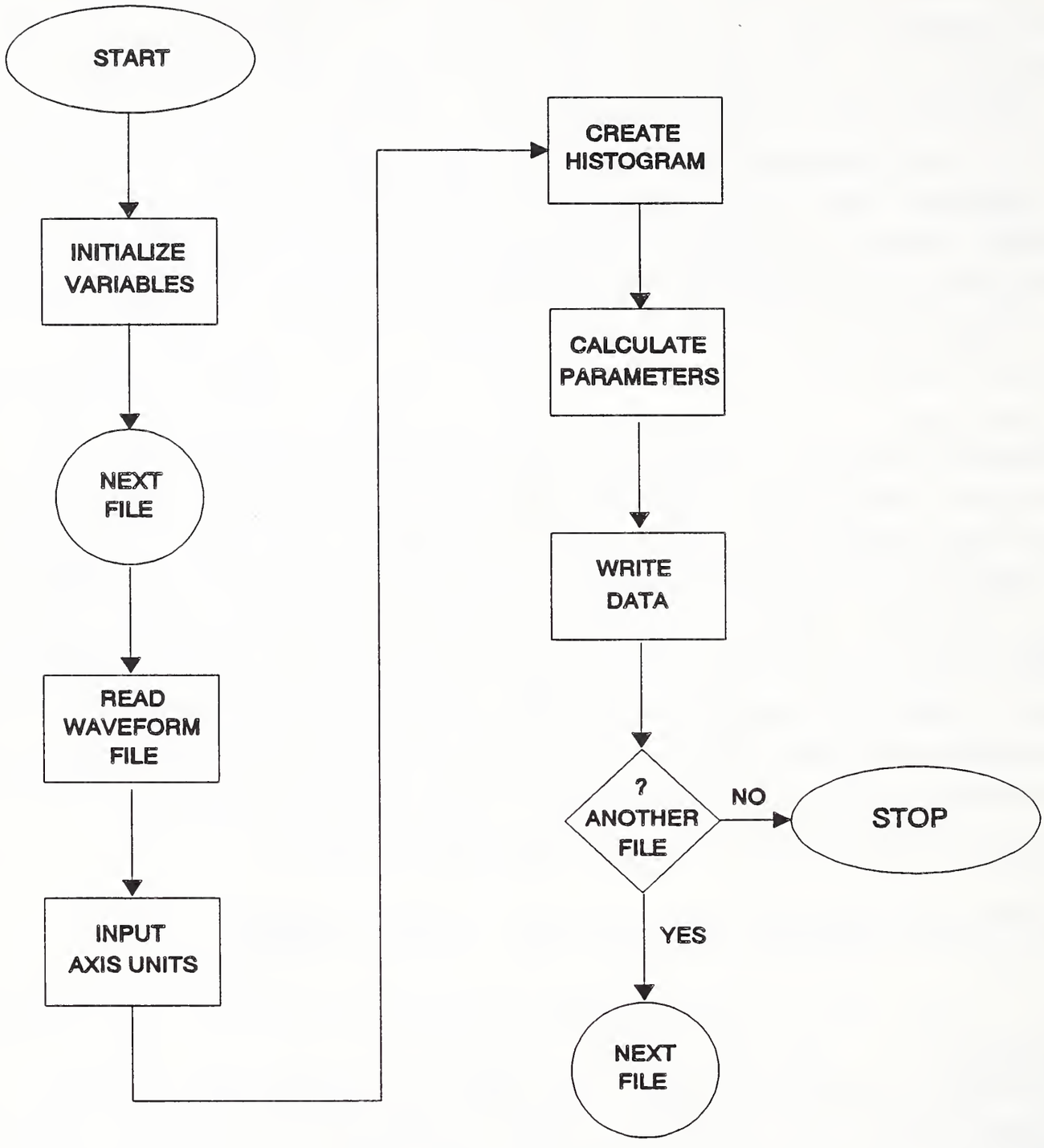

Figure 8. PULS_PARAMS flow chart.

4. percent overshoot; percent undershoot;
5. the second transition durations and pulse duration for impulse or square waveforms;
6. the number of bins used to create the histogram;
7. the 0 percent and 100 percent data values; and
8. the number of data occurrences at 0 percent and 100 percent.

It also displays the first, last, maximum, and minimum data values.

The first step in calculation of these parameters is finding an appropriate histogram. There should be roughly 10 points in either the 100 percent or 0 percent bin for an impulse-like waveform and 10 points for both the 0 percent and 100 percent bins for a step-like signal. The program will decrease the number of histogram bins until there are enough data occurrences in the 0 percent bin. The 0 percent bin is the only one checked by this program. Once the automatic bin calculation is done, you will be asked if the histogram is acceptable. Reasons for rejecting the histogram include an insufficient number of points at 100 percent or an inadequate y-axis resolution. If for any reason the histogram is unacceptable, you can override the automatic bin calculation and enter in the number of bins you want.

The second step is to define the 0 percent and 100 percent values. If the pulse waveform is negative-going, the 100 percent value becomes the 0 percent value and the 0 percent value becomes the 100 percent value. Be aware of this when selecting the definitions. Your options for 0 percent are:

1. value of the first point in the waveform;
2. value of the last point in the waveform;
3. the minimum value in the waveform;
4. the 0 percent value found by the histogram; or
5. you may input a value from the keyboard.

Your options for 100 percent are:

1. value of the first point in the waveform;
2. value of the last point in the waveform;
3. the maximum value in the waveform;
4. the 100 percent value found by the histogram; or
5. you may input a value from the keyboard.

If the waveform is step-like or square, you will want to pick the values found by the histogram (option 4) for both 0 and 100 percent. If the waveform is impulse-like, you will want to pick the value found by the histogram (option 4) for 0 percent and the maximum waveform value (option 3) for 100 percent.

After you input the required level definitions, the program automatically continues with the calculation of the pulse parameters. When these calculations are finished, you will be asked to input a file name for data storage. These data are stored in ASCII format.

You will then be asked if you would like a hard copy print-out of the results. If you choose not to print the result at run time, you can obtain a hard copy later by running the "text_out" program. This program simply reads in an ASCII file and prints it. Just load and run the program and answer the questions given.

4.6 Support Programs

4.6.1 Jitter waveform generator -- (GAUSS)

This program is used to create a waveform that approximates the jitter distribution function of the measurement system. This function can be deconvolved from the DUT waveform data. The first step in using this program is to measure sigma (σ) of the jitter of the DUT. Sigma is the sum of the squared deviations of the measurements from their mean. This measurement is explained in Section A.4.4 of this manual.

After measuring the value of σ of the jitter, load and run GAUSS (see Fig. 9). You will be prompted to input values for the number of points in the waveform, the epoch of the waveform, and the value of σ (Section A.4.4) as needed. The number of points and the epoch of the jitter distribution function must be equal to the number of points and the epoch of the DUT waveform for deconvolution. To make the jitter distribution function waveform more like an impulse response, we create a Gaussian curve with unit area. If you choose other than unit area, you will be prompted to input an amplitude value. When the waveform has been calculated, you will be asked for a file name for data storage. The result is stored, and the program ended. Again, the data are stored in binary data format and the file written is compatible with GRAPH_DATA.

4.6.2 Waveform math operations utility (MATH_OPS)

The MATH_OPS program allows you to perform some basic operations on the waveforms. These operations are: integration, differentiation, time shifting, and constant arithmetic operations of addition, subtraction, multiplication, and division (see Fig. 10). This program is similar in style to the

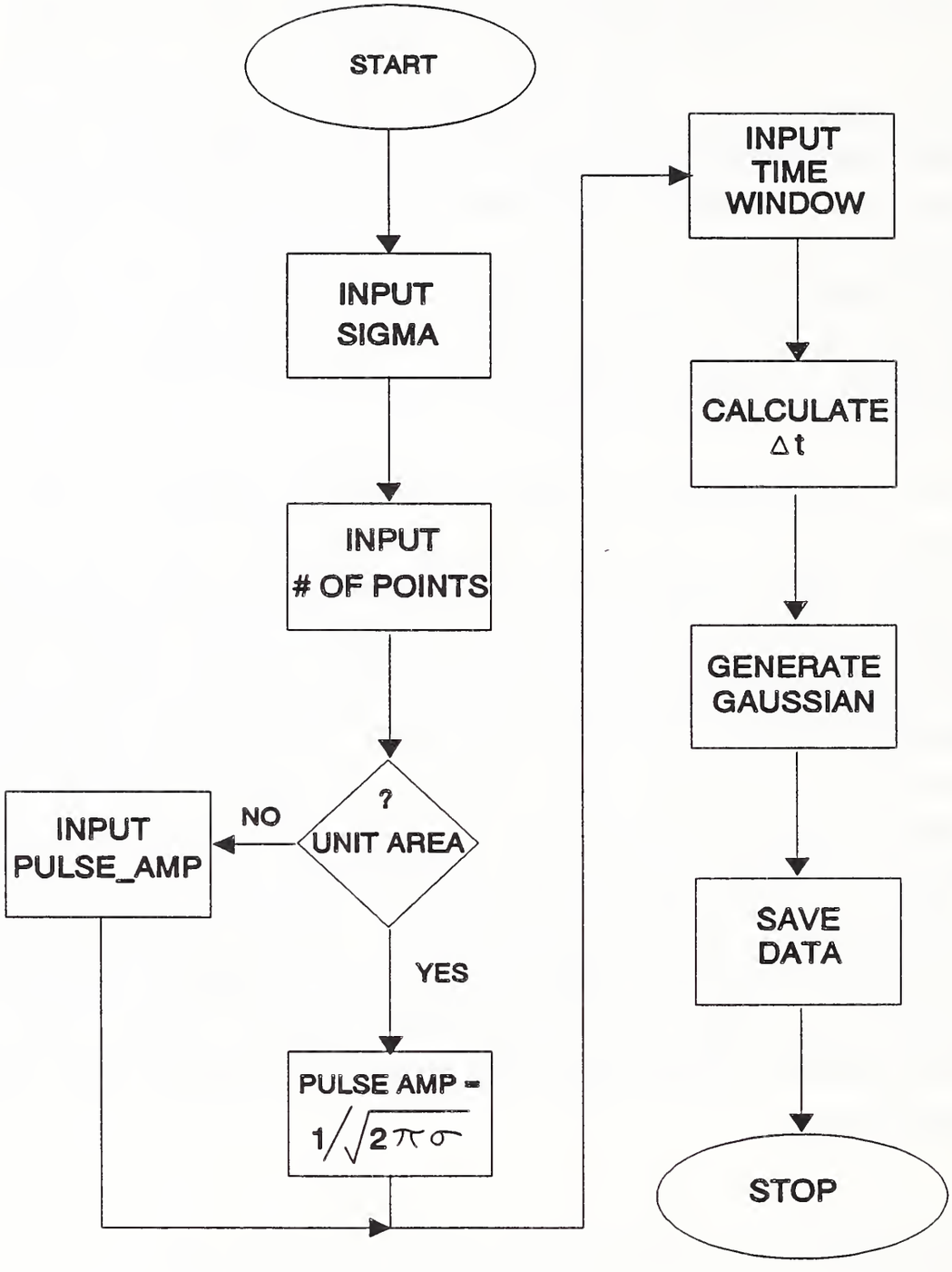

Figure 9. GAUSS flow chart.

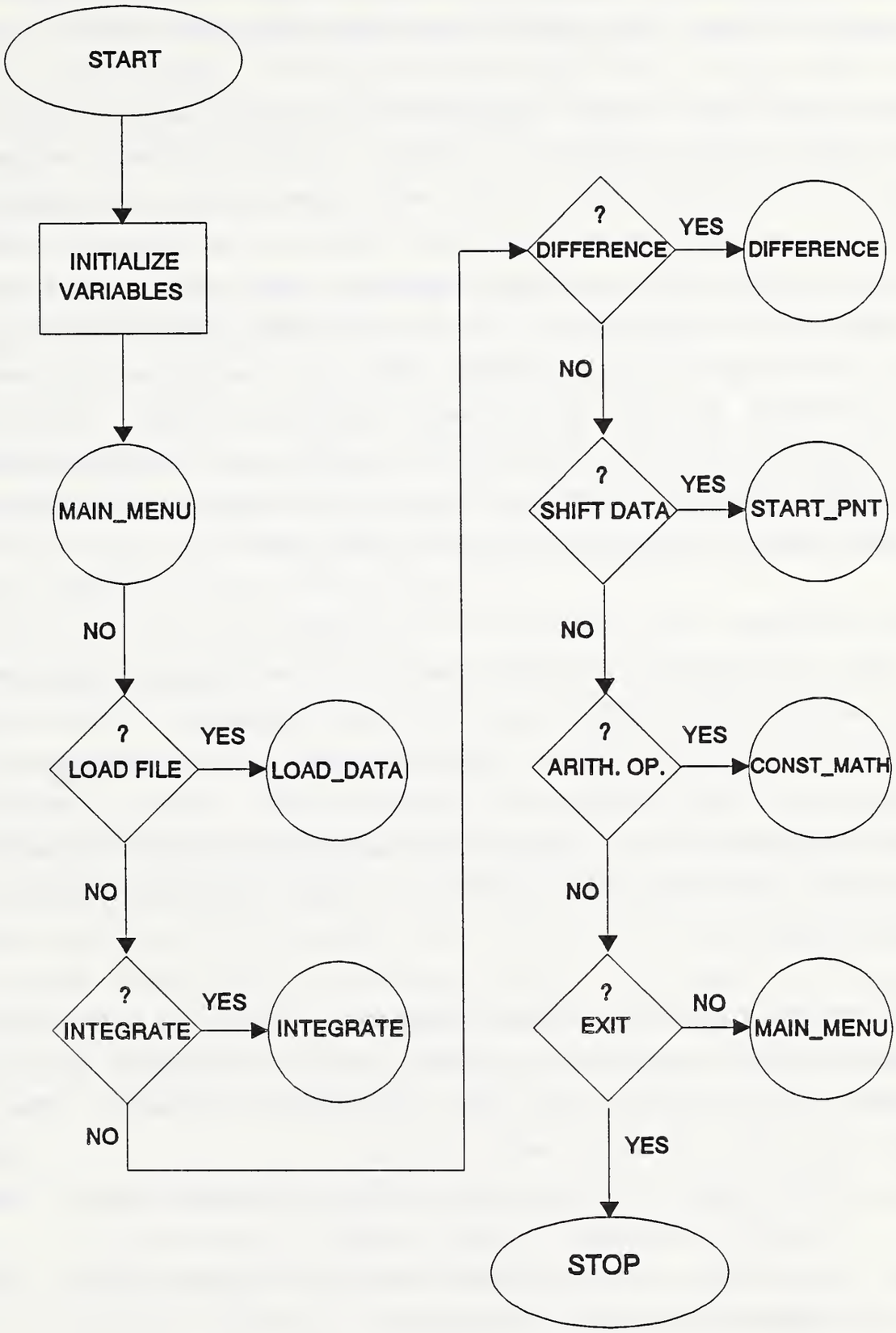

Figure 10. MATH_OPS flow chart.

acquisition program since it is menu driven. To perform one of the above operations, press the appropriate softkey. To leave the program, select the "Exit" key from the main menu.

The first step after loading and running this program is to use the "Load Data" softkey. If you fail to do this before attempting to perform any of the math operations, the program will display a message telling you there is no file in memory and return you to the main menu so that you may load in a file. After every math operation, the file in the program memory is purged (this does not imply that data have been lost since the data are stored in one of the mass storage units). This means you will have to load a file before beginning a math operation. This program reads and writes files in binary data format and the file structure is compatible with GRAPH_DATA.

The program will perform the corresponding function when you press the "Integrate" or "Differentiate" softkeys. Because the data on which this program operates are discrete, rather than continuous, the integration and differentiation are actually summation and first difference respectively. After completing the selected operation, the program requests a file name and a disk for data storage, the data are saved and the program returns to the main menu.

To access the time shift operation, which is useful for comparing multiple waveforms, press the "Time Shift" key from the main menu. The program will then display a new menu. The options available in this menu are to

1. start the array at the index corresponding to the maximum voltage value;
2. start the array at the index corresponding to the minimum voltage value; or
3. input the starting index number from the keyboard.

By aligning the waveforms according to a particular waveform feature you will be able to easily discern differences and similarities. After you select the required starting point, the data are rearranged. You are then asked for a file name and disk drive for data storage, the data are stored, and the program returns to the main menu. To see the rearranged waveforms, use GRAPH_DATA.

If you want to add, subtract, multiply, or divide the y-axis data by a constant, press the "+-*/ Y" menu key. You will then have the following softkey menu options: add, subtract, multiply, and divide. After selecting one of the options, input the value for the constant; the program then performs the required operation. Then when asked input the file name and designate a disk drive for data storage, the data are then saved, and the program returns to the main menu.

To leave this program, press the "Exit" key.

4.6.3 Graphics Support (GRAPH_DATA)

The AWAMS package includes an up-to-date version of the NIST-written graphics support package, GRAPH_DATA. GRAPH_DATA has been modified to include the ability to draw a histogram plot on the same graph with a waveform plot. In order to include the histogram plot with the waveform plot, you must follow a number of steps in sequence. It is easiest to follow this procedure while you are using GRAPH_DATA and actually performing the key presses.

1. Load the desired file or select it for plotting if it is already in GRAPH_DATA.
2. After selecting the required file, go to the "Auto Scale" option and rescale the graph to this data set. Now is the time to do any manual scaling you require. This is done in the "Edit Background" menu.

3. Select the "Edit Background" option, press the "Graph Type" softkey, and select the "Histogram" option.
4. Return to the "Edit Data" menu and select the "Data Math" key and then the "User" key (access the "User" key by pressing the shift key on the keyboard and 9th softkey). When asked, type in "GD_HISTOGRM". This subroutine operates in the same way as the histogram routine in the pulse parameters program, see Ref. [7] for details.
5. Finally, select the curve you just created for plotting and return to the main menu.

It is essential that you select only one file for both plotting and histogram generation as the results will be quite confusing if you do not. At this point, *do not re-scale the graph*. The scaling of the axes of the histogram graph is dependent on the position of the waveform data graph. Using "Auto Scale" or redefining the graph type will cause an odd looking result. However, you may add labels, change pen color, or any other operation that does not rescale the graph.

GRAPH_DATA has many other features, but these are not directly related to the AWAMS and are therefore not documented here.

5.0 SYSTEM CONSIDERATIONS

For details on computer, computer peripherals, and oscilloscope operation, see the corresponding operation manuals listed in Section A.3.

5.1 System Changes

The AWAMS, as delivered, will not need reconfiguring after installation. However, if you decide to change the device bus addresses or change the system hardware, you will also need to change the software to match. Feel free to change bus addresses, user messages, default values, etc. However, please contact the software author(s) *before* making any changes to the algorithms to maintain consistency between the NIST Automatic Waveform Analysis and Measurement System (AWAMS) and the AWAMS.

5.2 Acquisition Setup

1. When you acquire a waveform by the computer, the voltage scale seen on the oscilloscope screen is not necessarily the voltage scale acquired. For example, if the signal you measure is nominally 300 mV and the oscilloscope is set at 10 mV per division and the offset is set to center the signal, you will see a clipped waveform on the screen of the oscilloscope, but the acquired waveform will not be clipped. The exception is when the signal exceeds the maximum voltage for the analog-to-digital converter. In that case, the acquired signal will be clipped since this voltage exceeds the maximum oscilloscope capability. **WARNING:** The maximum safe input voltage into any measurement channel or into the external trigger is ±2 Vdc ± ac peak. Input voltages that exceed this level may damage or destroy the sampling circuitry. Measurement of signals of this magnitude, or

because you may destroy the sampling heads of the oscilloscope. See [MUD 1] for details on the oscilloscope specifications. If you want to see a clipped version of the waveform after acquisition, use the manual scaling feature of GRAPH_DATA to display only the desired part of the waveform.

2. Every 4 ns, the oscilloscope will update the clock for the time axis. This event occurs at

$$16+4Nns,\ N=0,1,2... \qquad (10)$$

It is essential that your measurement does not include these events. External delay line(s) inserted in the trigger circuit may be required to meet this condition. We do not recommend inserting extra cables in the signal path because of losses in the lines and subsequent distortion of the signal. If you have set up the measurement correctly, this should be no problem for small epochs.

3. When positioning the DUT waveform on the oscilloscope screen, you want the beginning and ending parts of the waveform to visually have nearly zero slope. This condition gives the best results for any subsequent deconvolution. Usually, when the waveform is in this position, the voltage midpoint will be between the first and third graticules.

4. The FFT used by the AWAMS requires 2^n points, where n is an integer. Pick the epoch for measurement so that you can chose the number of points accordingly. For instance, select a 2-ns epoch and then choose 2048 points.

5. Increasing the number of points and/or the number of averages in the measurement increases the acquisition time (Section 4.2). It takes approximately 15 minutes to acquire

(Section 4.2). It takes approximately 15 minutes to acquire a waveform with 1024 points and 2048 averages. Choose the maximum number necessary for each measurement.

6. The acquisition program checks for many error conditions. However, if you mistakenly acquire from the wrong sampling channel, the data will not be what you want. If you get a large number of time calibration factors after using the TCAL feature of the ACQUIRE program, this is likely the problem. Check your setup and try again.

5.3 Oscilloscope and Computer

1. For a complete list of the environmental and operating requirements of the oscilloscope and computer see [MUD {1},{8}]

2. Please wear the grounding wrist strap when operating the oscilloscope. For a complete list of precautions for avoiding electrostatic discharge damage to the oscilloscope, see [MUD {1}].

3. The oscilloscope's input connectors are precision 3.5-mm connectors. Although SMA and the precision 3.5-mm connectors appear to mate well, a bad SMA can ruin a good 3.5-mm connector. Please gauge all SMA connectors before coupling to the oscilloscope to verify that they will not harm the input connectors. To increase the lifetime and maintain precision of the connectors, you must use the connectors correctly. Please read [MUD {1}] for care and handling.

6.0 REFERENCES

1. A.V. Oppenheim and R.W. Schafer, Discrete Time Signal Processing, Prentice-Hall, Englewood Cliffs, NJ, 1989.

2. A.V. Oppenheim and A.S. Willsky, Signals and Systems, Prentice-Hall, Englewood Cliffs, NJ, 1983.

3. Taken from "Minimizing the effects of record truncation discontinuities in waveform deconvolutions" by N.G. Paulter and R.B. Stafford, with permission from N.G. Paulter. Submitted to the Institute of Electrical and Eelectronics Engineers for publication.

4. W.L. Gans and N.S. Nahman, "Continuous and Discrete Fourier Transforms of Steplike Waveforms," IEEE Trans. Instrum. Meas., IM-31, p.97, 1982.

5. N.S. Nahman and M.E. Guilluame, "Deconvolution of time domain waveforms in the presence of noise," Nat. Bur. Stand. (U.S.) Tech Note 1047, 1981.

6. "IEEE Standard Pulse Terms and Definitions," IEEE Std 194-1977, The Institute of Electrical and Electronics Engineers, Inc., New York, NY, 1977.

7. "IEEE Standard on Pulse Measurement and Analysis by Objective Techniques," ANSI/IEEE Std 181-1977, The Institute of Electrical and Electronics Engineers, Inc., New York, NY, 1977.

8. W. L. Gans, "The measurement and deconvolution of time jitter in equivalent-time waveform samplers," IEEE Trans. Instrum. Meas., IM-32, p. 126, 1983.

7.0 GLOSSARY

bandwidth
: the upper frequency limit at which the oscilloscope's voltage response has decreased to 71 percent from its flat frequency response region.

continuous time
: all time values, however small the time increment. A continuous-time function is defined for all time.

convolution
: mathematically, it is the integration of the multiplication of one function and a time-shifted replica of a second function. A physical example is the interaction of an input waveform and a measurement system which can be described mathematically by a convolution integral.

deconvolution
: a mathematical process that allows the reconstruction of one of the two waveforms involved in a convolution. Knowledge of one of the waveforms involved in the convolution is necessary in order to perform a deconvolution.

discrete-time
: specific time values, fixed incremental time steps. A discrete-time function is defined only at specific time values.

equivalent-time sampling
: sampling of successive points along a periodically repeated waveform so that the sequence of sampled points may be used to reconstruct the waveform. For example, sampling one point from nonidentical locations on each of ten identical repeats of a waveform will allow a 10-point reconstruction of the waveform.

Fourier transform
: mathematical transformation of time data into its sinusoidally varying components, or the reverse process.

impulse response
: the temporal response of a given device to excitation by a unit-height delta function, or impulse.

interpolate
: a method to infer the value of a nonexistent datum, where that datum is located between known or measured data values.

jitter
: random synchronization errors between the trigger and its associated event.

sampling window (aperture)
: the time duration over which the sampling process occurs.

time aliasing
: a distortion in the time representation of a waveform caused by not satisfying the sampling criterion. Basically, the ends of the waveforms overlap and add, thus causing errors in these portions of the waveforms.

8. ACKNOWLEDGEMENTS

We thank Larry Tarr for his patience with and support of this project. Additionally, we thank Bill Kissick, Chriss Jones, Bill Gans, and Nancy Keogel Sauer for their contributions to this manual; and John Ladbury, Chriss Jones, and Bob Stafford for their inputs, insights, and hours of consultation during the software development. We also thank Galen Koepke for his help with the software. His advice, support, and subroutines were invaluable to this project.

APPENDIX A

A.1 Specifications

The following table lists the various pulse parameters, including their ranges and uncertainties, that are measured with the AWAMS.

PARAMETER	RANGE	TYPICAL LIMITS OF UNCERTAINTY
Pulse Baseline (0% level)	±500 mV	±(0.5% + 3 mV)
Pulse Topline (100% level)	±500 mV	±(0.5% + 3 mV)
Pulse Amplitude	±500 mV	±(0.5% + 3 mV)
Pulse First Transition Duration (Rise Time)	10 ps to 100 ns	±(0.5% + 3 ps)
Pulse Second Transition Duration (Fall Time)	10 ps to 100 ns	±(0.5% + 3 ps)
Pulse Duration (between 50% levels)	10 ps to 100 ns	±(0.5% + 3 ps)

A.2 AWAMS Hardware

The following list provides an inventory of the hardware presently provided with the AWAMS. A brief description of these components is also given. However, for detailed information, please see the manufacturer users's manuals listed in Sec. A.3. We use trade names to specify the equipment used in this system and no endorsement by the National Institute of Standards and Technology is implied. Similar products by other manufacturers may work as well or better.

Digitizing oscilloscope

The HP Model 54120T digitizing oscilloscope provides 20 GHz bandwidth, full IEEE 488 programmability, time-domain reflectometry capability, and waveform arithmetic operations.

Computer/instrument controller

The HP 9000/300 Microcomputer operates at 16.6 MHz, has 8 megabytes of RAM, and IEEE 488, LAN, and RS232C interfaces.

Disk Drives

The HP 9127A Disk Drive uses 5.25-inch diameter, magnetic storage, flexible disks.

The HP 9122C has 2-megabyte data storage capacity and uses double-sided, 3.5-inch diameter floppy disks.

The HP 7957B is a hard disk drive unit and has 81-megabyte of data storage capacity.

Printers

The HP ThinkJet and HP PaintJet printers are used with the AWAMS.

A.3 Manufacturers' Users' Documentation

The following is a list of the manufacturers' users' documentation. Familiararity with these manuals is essential to the proper operation of the AWAMS.

1. "HP 54120 User Documentation (Installation/Operation Manual)," HP Manual Part Number 98613-90000, Hewlett-Packard Company, U.S.A., 1987.

2. "HP 54121T Digitizing Oscilloscope Programming Reference," HP Manual Part Number 54121-90907, Hewlett-Packard Company, U.S.A., 1989.

3. "Using the BASIC 5.0/5.1 System, HP 9000 Series 200/300," HP Part Number 98613-90000, Hewlett-Packard, U.S.A., 1988.

4. "BASIC 5.0/5.1 Interfacing Techniques, Vol. 1: General Topics, HP 9000 Series 200/300 Computers," HP Part Number 98613-90022, Hewlett-Packard Company, U.S.A., 1987.

5. "Basic Language Reference Volume 1 A-N," HP Part Number 98613-90052, Hewlett-Packard Company, U.S.A., 1989.

6. "Basic Language Reference Volume 1 O-Z," HP Part Number 98613-90052, Hewlett-Packard Company, U.S.A., 1989.

7. "Basic 5.0/5.1 Programming Techniques Volume 1 : General Topics," HP Part Number 98613-90813, Hewlett-Packard Company, U.S.A., 1988.

8. "Installation Reference, HP 9000 Series 300 Computers," HP Part Number 9856-90000, Hewlett-Packard Company, U.S.A., 1988.

9. "Getting Started with Your HP 9127A Disc Drive," HP Manual Part Number 09127-90000, Hewlett-Packard Company, U.S.A., 1989.

10. "Getting Started with Your HP 9122C Disc Drive," HP Manual Part Number 09122-90901, Edition 2, Hewlett-Packard Company, U.S.A., 1988.

11. "HP 7957B, HP 7958B, and HP 7959B Disc Drives," HP Manual Part Number 07959-90901, Hewlett-Packard Company, U.S.A., 1988.

12. "Personal Printer ThinkJet, Owner's Manual," HP Manual Part Number 02225-90031, Hewlett-Packard Company, Singapore, 1987.

A.4 Measurement Setup and Example Procedures

A.4.1 General

Please read the section titled "Handling and Care of the Precision Connectors" [MUD{1}] before using the oscilloscope.

1. Clean all connectors on the oscilloscope, the DUT, the voltage calibration standard, the time calibration standard, and any required adapters, filters, and cables.
2. Connect the 3.5-mm shorts to all of the oscilloscope channels and the trigger input. Run the vertical calibration utility of the oscilloscope [MUD{1}].
3. You may use any of the four sampling channels for the measurement; this is represented as "Chan X" in Figs. A1 and A2. The asterisks in Figs. A1 and A2 indicate the equipment supplied as part of the AWAMS.
4. The solid lines in Figs. A1 and A2 represent the DUT connections, the dotted lines the time calibration standard connections, and the dashed lines the voltage calibration connections. The test equipment marked with an "*" in Figs. A1 and A2 are the only pieces that are provided as part of the AWAMS. You must provide all other test equipment.
5. Record the time, the date, and the room temperature and humidity before beginning data acquisition. These values should be recorded periodically throughout the test.

A.4.2 Pulse generator, procedure 1

This procedure is used for measuring the Tektronix S-52 pulse generator output; this pulser will be referred to as P1. NIST has received many requests to the measure the output of P1 and, consequently, has developed procedures for measuring its

CALIBRATION SETUP FOR P1

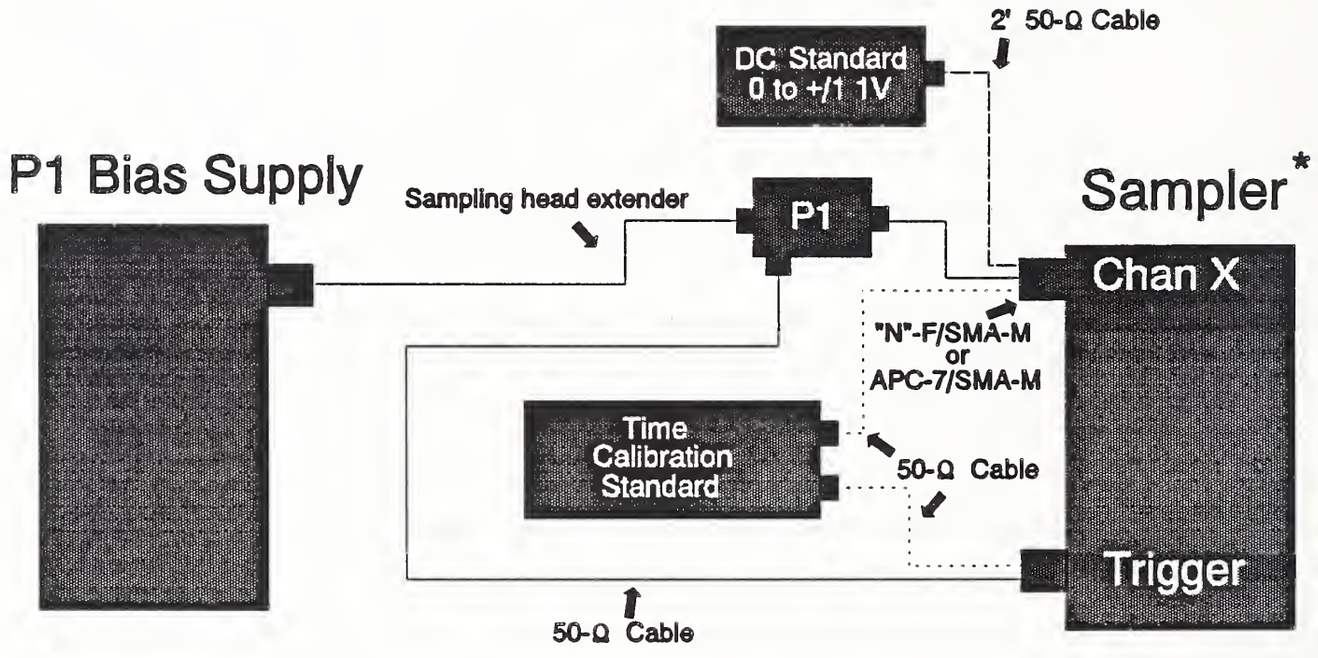

Figure A1. Pulse generator measurement setup 1.

CALIBRATION SETUP FOR P2

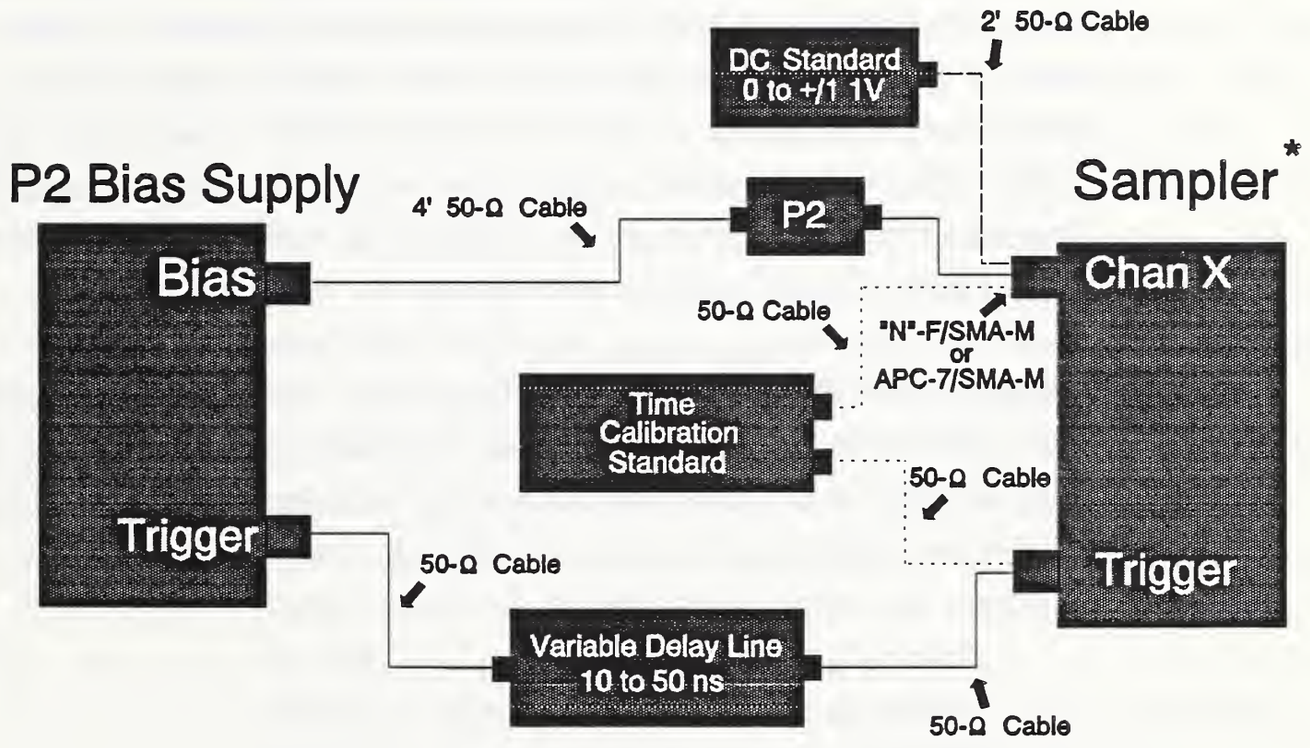

Figure A2. Pulse generator measurement setup 2.

output. However, this does not imply any preference, by NIST, for this product.

1. Load and run the ACQUIRE program. You should determine and set the acquisition parameter values for each menu before acquiring any data. Procedures for determining these values follow. Once the menus have been set up, you will not need to redo them. You will acquire nine or more data sets. A data set consists of one DUT waveform, one time-calibration waveform, and one group (see VCAL description) of voltage-calibration factors. You may acquire these three components in any order but each set must be completed before beginning another. Once you choose an order of acquisition, maintain that order throughout the test. For example, first acquire the time-calibration data, then the voltage-calibration data, and finally, the DUT data.

2. Connect the P1 as shown in Fig. A1. Press the waveform menu key and select the manual setup option. View the DUT waveform on the oscilloscope screen and manually position the waveform as required.
   Typical P1 settings are:
   Channel 4;
   Volts per Division: 50 mV;
   Offset: 215.00 mV;
   Attenuation: 1;
   Time per Division: 200 ps;
   Delay: 81.275 ns;
   Delay Reference: center;
   Points: 1024;
   Averages: 2048;
   Trigger Level: 300 mV;
   Trigger Slope: positive (this is not one of the ACQUIRE program menu selections).

These are typical settings; yours may be different. Write down the oscilloscope settings that you use. Make certain that the time window does not have one of the 4-ns boundaries described in Sec. 5.2. If the epoch does have a 4-ns boundary, you may need to use an external delay line to reposition the waveform. Check the minimum and the maximum voltages that span the waveform's voltage range; this will be used in the subsequent voltage calibration. Once the oscilloscope has been setup, return to the main menu.

3. Connect your time calibration standard as shown in Fig. A1. Press the TCAL menu key and then the manual setup key. View the waveform and establish the required oscilloscope settings. The first mid-point crossing of the time calibration waveform should occur before the location of the first transition of the pulse waveform. **The time/div and the delay settings of the oscilloscope must be the same as the those for DUT. Use external delay lines to position the calibration waveform.** You may change the volts per division and offset settings as needed. Typical time calibration settings are:

   Frequency: 5.0×10^9 Hz;
   Volts per division: 1.0 mV;
   Offset: 0.0 mV (as needed to center the waveform);
   Number of sliding average point pairs: 7;
   Number of least squares points: 20;
   Slope: positive;
   Save the waveform: no;
   Time calibration signal trigger level: 150 mV.

   Record the time calibration settings. Return to the main menu.

4. Connect your DC standard as shown in Fig. A1. Press the VCAL key and then the manual setup key. **The volts**

per division and offset values must be the same as those for the DUT. Other settings will invalidate the voltage-calibration data. Determine the number of voltage steps and the voltage increment needed to span the waveforms voltage range (previously determined). Typical settings are:

Voltage intervals: 7;
Averages: 128;
Minimum or starting voltage: 0.05 mV;
Voltage step size: 0.05 mV.

Record the voltage calibration settings.

5. Acquire and record the nine sets of data.
6. When you have collected nine data sets, exit the ACQUIRE program.
7. Measure the sigma for the jitter impulse response. The procedure for measuring the jitter is outlined in Sec. A.4.4 of this Manual.
8. Create the jitter impulse response using the GAUSS program.
9. Calibrate all nine waveforms using the FIXACQ program and the corresponding calibration data.
10. Deconvolve the jitter function from all nine calibrated waveforms using the DECON_NIST program.
11. Deconvolve the system impulse response, SYS_RESP, from all nine calibrated, jitter-deconvolved waveforms using the DECON_NIST program. SYS_RESP is the name of the file that contains the system impulse response for the oscilloscope.
12. Run the PULS_PARAMS program on all nine of the calibrated, jitter-deconvolved, system-deconvolved waveforms. These are the corrected waveforms.
13. Calculate the mean and standard deviation of the following parameters: pulse amplitude (not the peak-to-peak value), 10-90% transition duration, 20-80% transition duration, percentage overshoot, and the

percentage undershoot. (Do these calculations for all of the pulse parameter data that you have generated.)

14. Determine which of the nine corrected waveforms has parameter values closest to the mean values calculated above; this is the representative waveform. Use this representative waveform for the report plots.
15. Plot the representative waveform.
16. Calculate the uncertainties in the representative waveform's pulse parameters. You may use the limits listed in Sec. A.1 for these calculations.
17. Generate the test report.

A.4.3 Pulse generator, procedure 2

This procedure is used for measuring the Hewlett-Packard 1106A and 1106B pulse generator outputs; these pulsers will be referred to as P2. NIST has received many requests to the measure the output of P2 and, consequently, has developed procedures for measuring its output. However, this does not imply any preference, by NIST, for this product.

1. Load and run the ACQUIRE program. You should determine and set the acquisition parameter values for each menu before acquiring any data. Procedures for determining these values follow. Once the menus have been set up, you will not need to redo them. You will acquire nine or more data sets. A data set consists of one DUT waveform, one time-calibration waveform, and one group of voltage-calibration factors. You may acquire these three components in any order each set must completed before beginning another. Once you choose an order of acquisition, maintain that routine throughout the test. For example, first acquire the time-calibration data, then the voltage-calibration data, and finally, the DUT data.

2. Connect P2 as shown in Fig. A2. Press the waveform menu key and then select the manual setup option. View the DUT waveform on the oscilloscope screen and manually position the waveform as required.
Typical P2 settings are:

Channel 4;
Volts per division: 50 mV;
Offset: 235.00 mV (as required to center the waveform);
Attenuation: 1;
Time per division: 200 ps;
Delay: 33.100 ns;
Delay Reference: center;
Points: 1024;
Averages: 2048;
Trigger Level: 400 mV;
Trigger Slope: positive (this is not one of the ACQUIRE program menu selections).

These are typical settings; yours may be different. Write down the oscilloscope settings that you use. Make certain that the time window does not have one of the 4-ns boundaries described in Sec. 5.2. If the epoch does have a 4-ns boundary, you may need to use an external delay line to reposition the waveform. Check the minimum and the maximum voltages that span the waveform's voltage range; this will be used in the subsequent voltage calibration. Once the oscilloscope has been set up, return to the main menu.

3. Connect your time calibration standard as shown in Fig. A2. Press the TCAL menu key and then the manual setup key. View the waveform and establish the required oscilloscope settings. The first mid-point crossing of the time calibration waveform should occur before the

location of the first transition of the pulse waveform. **The time per division and the delay settings of the oscilloscope must be the same as the those for DUT. Use external delay lines to position the calibration waveform.** You may change the volts per division and offset settings as needed. Typical time calibration settings are:

Frequency: 5.0×10^9 Hz;
Volts per division: 1.0 mV;
Offset: 0.0 mV (as needed to center the waveform);
Number of sliding average point pairs: 7;
Number of least squares points: 20;
Slope: positive;
Save the waveform: no;
Time calibration signal trigger level: 150 mV.

Record the time calibration settings. Return to the main menu.

4. Connect your DC standard as shown in Fig. A2. Press the VCAL key then the manual setup key. **The volts per division and offset values must be the same as those for the DUT.** Other settings will invalidate the voltage calibration data. Determine the number of voltage steps and the voltage increment needed to span the waveforms voltage range (previously determined). Typical settings are:

   Voltage intervals: 7;
   Averages: 128;
   Minimum or starting voltage: 0.05 mV;
   Voltage step size: 0.05 mV.

   Record the voltage calibration settings.
5. Acquire and record the nine sets of data.
6. When you have collected nine data sets, exit the ACQUIRE program.

7. Measure the sigma for the jitter impulse response. The procedure for measuring the jitter is outlined in Sec. A.4.4 of this Manual.
8. Create the jitter impulse response using the GAUSS program.
9. Calibrate all nine waveforms using the FIXACQ program and the corresponding calibration data.
10. Deconvolve the jitter function from all nine calibrated waveforms using the DECON_NIST program.
11. Deconvolve the system impulse response, SYS_RESP, from all nine calibrated, jitter-deconvolved waveforms using the DECON_NIST program. SYS_RESP is the name of the file that contains the system impulse response for the oscilloscope.
12. Run the PULS_PARAMS program on all nine of the calibrated, jitter-deconvolved, system-deconvolved waveforms. These are the corrected waveforms.
13. Calculate the mean and standard deviation of the following parameters: pulse amplitude (not the peak-to-peak value), 10-90% transition duration, 20-80% transition duration, percentage overshoot, and the percentage undershoot. (Do these calculations for all of the pulse parameter data that you have generated.)
14. Determine which of the nine corrected waveforms has parameter values closest to the mean values calculated above; this is the representative. Use this representative measurement waveform for the report plots.
15. Plot the representative waveform.
16. Calculate the uncertainties in the representative waveform's pulse parameters. You may use the limits listed in Sec. A.1 for these calculations.
17. Generate the test report.

A.4.4 Jitter Measurement

1. Use the same oscilloscope settings as those for the DUT. Record the time and the room temperature and relative humidity.
2. Select the "Delta V" oscilloscope menu key. Turn the voltage markers on and select the measurement channel for both markers.
3. Press the "Auto Level Set" key. Select the "Preset Levels = 50%-50%."
4. Record the displayed 50% amplitude value.
5. Select the "Histogram" oscilloscope menu key, "Time Histogram," and then "Window."
6. Using the oscilloscope knob, set marker number 2 to the 50% voltage value.
7. Use the oscilloscope knob to set marker 1 to one voltage increment above the marker-2 voltage position. This is usually approximately 1.5 mV. Record the voltage values of each marker.
8. Select the "Acquire" oscilloscope key and then set the "Number of Samples" to 500.
9. Press the "Start Acquire" key.
10. When the oscilloscope has finished the measurement, press the "Results" key and then the "Sigma" key.
11. Record the displayed value of sigma.
12. Repeat steps 8 through 11 nine times.
13. Calculate the mean and standard deviation of the nine recorded values. Use the mean value for the input to the GAUSS program.

APPENDIX B (Software source code listings)

This appendix contains the source code listing of the programs *ACQUIRE*, *DECON_NIST*, *FIXACQ*, *PULS_PARAMS*, *GAUSS*, *GD_HISTOGRAM*, *MATH_OPS*, *text_out* that are used in the AWAMS.

```
100 ! RE-STORE "ACQUIRE:,1400"
102   !
104   !
106   COM /Interrupts/ INTEGER Intr_prty
108   COM /Sys/ Sys_id$[10]
110   COM /Sys_msi/ Msi_id$[20]
112   !
114   OUTPUT KBD USING "K,#";"SCRATCH KEY"        !ERASE SOFT KEYS
116   CONTROL KBD,15;0        ! sets the colors of the soft keys
118   CONTROL KBD,2;1
120   !
122   Intr_prty = 1
124   CALL Cal_prog
126   !
128   OUTPUT KBD USING "K,#";"LOAD KEYE"       ! returns the typing aid keys
130   PRINT TABXY(1,5);"End of program. Enter 'RUN' to repeat."
132   !
134   END           ! end of stub that calls the Cal_prog subprogram
136   !
138   !_____________________________________________________________________
140   !
142   SUB Cal_prog
144     !
146 Date_line:     !
148     !    ---------------------------------
150     !   Last modified on 20 MAY 91 at 12:00
152     !    ---------------------------------
154     !
156     !
158     !    This is version 2.0
160     !    Program  by S. M. Chesnut.
162     !    This program uses all kinds of stuff written by Galen Koepke
164     !    including the following: Menu_scroll, Select_disk,Enterfilename,
166     !    File_menu, Pause_key_on, Errortrap, and Data_to_disk_r.
168     !    Many thanks go to him for the use of these sub-programs.
170     !    Without his help and support, this program would not have
172     !    possible.
174     !    The upgrade to version 1.0 consisted of adding the ability
176     !    to access HFS disks and TIMEOUT, trigger, and/or internal
178     !    step generator error checks.
180     !
182     ! = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
184     !  Main Program
186     ! = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
188     !
190       OPTION BASE 1
192       DEG
194       KBD CMODE ON
196       PRINTER IS CRT
198       CLEAR SCREEN
200       OFF KEY
202       Version$ = "HP 320-12/20/89 version 1.0 subprogram "
204       !
206       !
208       GOSUB Init_variables
```

```
210       GOSUB Init_graphics
212       GOSUB Init_mnemonics
214       GOSUB Load_averages
216       GOSUB Main_cont
218       !
220       !
222 Exit:   ! !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
224       !
226       CLEAR SCREEN
228       LOCAL 707
230       !
232       IF Data_set_count MOD 8<>0 THEN
234         PRINTER IS PRT
236         PRINT CHR$(12)
238         PRINTER IS CRT
240       END IF
242       !
244       GCLEAR
246       GINIT
248       PRINT Enh_off$
250       OFF KBD
252       KBD CMODE OFF
254       SUBEXIT      ! exit the subprogram
256       !
258       ! ==========================================================
260       !  INITIALIZATION
262       ! ==========================================================
264       !
266 Init_variables:!
268       !
270       COM /Interrupts/ INTEGER Intr_prty
272       COM /Sys_msi/ Msi_id$[20]
274       COM /Sys/ Sys_id$[10]
276       COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
278       COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
280       COM /Hue/ Rev_vid$[1],Enh_off$[1],Underline$[1]
282       COM /Hue/ Red$[1],Orange$[1],L_blue$[1]
284       COM /Mnu/ INTEGER Interrupted,Which,T_which,V_which,Allowd$(12)[10]
286       COM /Mnu/ INTEGER Stp
288       COM /Scope/ REAL Time_per_div,Volts,Trange,Vrange,Dly
290       COM /Scope/ REAL Probe_fac,Offs,Trig,Atten
292       COM /Scope/ INTEGER Aver,Pnts,Chnnl
294       COM /Scope/ Type$[30],Refer$[14],@Scope,Mode$[30]
296       COM /Tcal_vals/ INTEGER Zero_x,Ls_prs,Slope,T_aver,REAL Freq
298       COM /Tcal_vals/ Slpe$[10],Save$[10],REAL Tc_off,Tc_volt,T_trig
300       COM /Vcal_vals/ INTEGER V_aver,Interval,REAL V_step,V_min
302       !
304       DIM Version$[80]
306       !
308       Sys_id$=SYSTEM$("SYSTEM ID")
310       Msi_id$=SYSTEM$("MSI")
312       !
314       Last1=0
316       IF Which=0 THEN
318         Which=1
320         T_which=1
322         V_which=1
324         GOSUB Start_vals
```

```
326                 !value for the initial run.
328                 !otherwise stored in COM/mnu, COM/Scope,
330                 !COM/Tcal_vals, and COM/Vcal_vals
332       END IF
334       PEN 1
336       Beep_flag=0
338       Stp=0
340       Info_screen$="n"
342       Local_prty=!ntr_prty
344       Paused=1.3
346       Mode$="TRIGGERED"
348       !
350       RETURN
352       !
354   !
356   !*************************************************************************
358   !
360 Init_mneumonics:   !
362       !
364       ! These are used to change the color with a PRINT or DISP statement
366       !
368       Red$=CHR$(137)
370       Orange$=CHR$(138)
372       L_blue$=CHR$(140)
374       Rev_vid$=CHR$(129)
376       Enh_off$=CHR$(128)
378       Underline$=CHR$(132)
380       !
382       RETURN
384       !
386   !
388   !*************************************************************************
390   !
392 !nit_graphics:!
394       GCLEAR
396       GINIT
398       Y_gdu_max=100*MAX(1,1/RATIO)
400       X_gdu_max=100*MAX(1,RATIO)
402       VIEWPORT 0,X_gdu_max,0,Y_gdu_max
404       WINDOW 0,1000,0,1000
406       GRAPHICS ON
408       ALPHA ON
410       RETURN
412       !
414   !
416   !*************************************************************************
418   !
420 Start_vals:       ! set-up default values
422       !
424       !Waveform variables
426       !
428       Trange=5.E-9
430       Time_per_div=Trange/10
432       Dly=6.0E-8
434       Chnnl=2
436       Pnts=128
```

```
438       Aver=64
440       Vrange=4.00E-1
442       Volts=Vrange/8
444       Type$="AVERAGE"
446       Offs=1.90E-1
448       Atten=1
450       Refer$="    CENTER    "
452       Trig=1.00E-1
454       !
456       !Voltage calibration default values.
458       !
460       V_aver=64
462       Interval=8
464       V_step=.05
466       V_min=0.
468       !
470       !Time calibration default values.
472       !
474       Slope=1
476       Freq=5.E+9
478       Slpe$=" + SLOPE "
480       Ls_prs=1/(Time_per_div*Freq)*Pnts/100+1
482       Zero_x=2*Ls_prs+1
484       T_aver=64
486       Tc_volt=Volts
488       Tc_off=Offs
490       Save$="   NO    "
492       !
494       RETURN
496   !
498   !*************************************************************************
500   !
502 Load_averages:      !
504 Strt:  DATA "1"
506       DATA "2"
508       DATA "4"
510       DATA "8"
512       DATA "16"
514       DATA "32"
516       DATA "64"
518       DATA "128"
520       DATA "256"
522       DATA "512"
524       DATA "1024"
526       DATA "2048"
528       RESTORE Strt
530       READ Allowd$(*)
532       RETURN
534   !
536   !*************************************************************************
538   !
540   !*************************************************************************
542   !
544 Main_cont:       !
546       !
548       Interrupted=1
```

```
550       Intr_prty=Intr_prty+1
552       LOOP
554          IF Interrupted THEN GOSUB Main_menu
556          IF Stp THEN GOTO Ret1
558          ON KEY 9 LABEL "EXIT  PROGRAM",Local_prty+2 GOTO Ret1
560       END LOOP
562       !
564 Ret1:  OFF KEY
566       Intr_prty=Intr_prty-1
568       RETURN
570       !
572   !
574 Main_menu:  !
576       Interrupted=0
578       PRINT Rev_vid$;L_blue$
580       PRINT "Press the appropriate soft key.";Enh_off$
582       ON KEY 0 LABEL "WAVEFORM  MENU",Local_prty+1 CALL Wave
584       ON KEY 2 LABEL "  TCAL  MENU  ",Local_prty+1 CALL Tcal
586       ON KEY 4 LABEL "  VCAL  MENU  ",Local_prty+1 CALL Vcal
588       ON KEY 5 LABEL " PROGRAM INFO ",Local_prty+1 CALL Operator_info
590       RETURN
592   !
594   !*************************************************************************
596   !
598    SUBEND
600   !
602   !*************************************************************************
604   !
606    SUB Wave
608 Wave:   !
610       CLEAR SCREEN
612       OPTION BASE 1
614       DEG
616       KBD CMODE ON
618       PRINTER IS CRT
620       OFF KEY
622       GOSUB Init_vars
624       GOSUB Wave_scope
626       GOSUB Wave_cont
628       SUBEXIT
630    !
632    ! ********************************************************************
634    !
636 Init_vars: !
638       COM /Interrupts/ INTEGER Intr_prty
640       COM /Sys_msi/ Msi_id$[20]
642       COM /Sys/ Sys_id$[10]
644       COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
646       COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
648       COM /Hue/ Rev_vid$[1],Enh_off$[1],Underline$[1]
650       COM /Hue/ Red$[1],Orange$[1],L_blue$[1]
652       COM /Mnu/ INTEGER Interrupted,Which,T_which,V_which,Allowd$(*)
654       COM /Mnu/ INTEGER Stp
656       COM /Scope/ REAL Time_per_div,Volts,Trange,Vrange,Dly
658       COM /Scope/ REAL Probe_fac,Offs,Trig,Atten
660       COM /Scope/ INTEGER Aver,Pnts,Chnnl
```

```
662     COM /Scope/ Type$[30],Refer$[14],@Scope,Mode$[30]
664     COM /Slect/ INTEGER Last1,Tlast,Vlast
666     !
668     !
670     DIM Version$[80]
672     DIM Test$[160],Data_id$[40]
674     DIM Choice$[15],Ch$[1],Step$[50]
676     !
678     !
680     INTEGER Co_ords(12,4),Lwrlftx,Lwrlfty,Upperrtx,Upperrtty
682     INTEGER Num_of_choices,What(1)
684     INTEGER I,J,Pen,Knobcount
686     INTEGER Error_flag,Beep_flag,Local_prty,Valid,Datacount
688     INTEGER Filesize,Baddata,Endpoint,Print_val
690     INTEGER Yref,Temp,Item_cnt,Err_flg
692     !
694     !
696     REAL Data_entered,Data_set_count
698     REAL Yinc,Yor,Rtemp,Waveform(32767)
700     DIM Dp$[80]
702     DIM T$[52]
704     Local_prty=Intr_prty
706     Ftype$="BDAT"
708     Filesize=500
710     Last1=0
712     PEN 1
714     Beep_flag=0
716     Paused=1.5
718     !
720     RETURN
722 !
724 !*************************************************************************
726 !
728 Wave_scope:     !
730     !This returns the waveform menu/scope values to those in
732     ! the common block called Scope.
734     CALL Scope_init(Err_flg)
736     IF Err_flg THEN
738        Wave_intrpt=1
740        Interrupted=1
742        Err_flg=0
744        SUBEXIT
746     END IF
748     CLEAR 707 !clears the HPIB to the scope
750     OUTPUT @Scope;":TRIGGER:LEVEL "&VAL$(Trig)
752     IF Refer$="   CENTER   " THEN
754        OUTPUT @Scope;"TIMEBASE:REFERENCE CENTER"
756     ELSE
758        OUTPUT @Scope;"TIMEBASE:REFERENCE LEFT"
760     END IF
762     OUTPUT @Scope;":TIMEBASE:DELAY "&VAL$(Dly)
764     OUTPUT @Scope;":TIMEBASE:RANGE "&VAL$(Trange)
766     OUTPUT @Scope;"VIEW CHANNEL"&VAL$(Chnnl)
768     OUTPUT @Scope;":CHANNEL"&VAL$(Chnnl)&":OFFSET "&VAL$(Offs)
770     OUTPUT @Scope;":CHANNEL"&VAL$(Chnnl)&":RANGE "&VAL$(Vrange)
772     OUTPUT @Scope;":ACQUIRE:COUNT "&VAL$(Aver)
```

```
774       RETURN
776       !
778   !
780   !*************************************************************************
782   !
784 Graphs_ary300: !
786               !This routine fills the array used by the select_graphics
788               !routine. The array contains the coordinates for the
790               !rectangles drawn in select_graphics.
792               !if the data is all zeros, this is a null field and is
794               !used to make all rows of equal length.
796               !These coordinates are for the system 300 machines.
798 First: DATA 30,943,176,34  !1,1
800       DATA 372,943,176,34 !1,2
802       DATA 714,943,220,34 !1,3
804       DATA 30,830,176,35 !2,1
806       DATA 372,830,176,35 !2,2
808       DATA 714,830,220,35 !2,3
810       DATA 30,717,176,35  !3,1
812       DATA 372,717,176,35 !3,2
814       DATA 714,717,220,35 !3,3
816       DATA 30,604,176,35  !4,1
818       DATA 0,0,0,0        !4,2
820       DATA 0,0,0,0        !4,3
822       RESTORE First
824       READ Co_ords(*)
826       Num_of_choices=12    ! total number of on screen choices
828       Rowsize=3
830       RETURN
832       !
834   !
836   !*************************************************************************
838   !
840 Wave_cont:   !
842       !
844       OFF KEY
846       Wave_intrpt=1
848       Intr_prty=Local_prty+1
850       LOOP
852          IF Wave_intrpt=1 THEN
854             IF Sys_id$[1,4]="S300" THEN
856                CONTROL CRT,5;1
858                SEPARATE ALPHA FROM GRAPHICS
860                GOSUB Graphs_ary300
862             END IF
864          END IF
866          IF (Wave_intrpt=1) OR (Wave_intrpt=3) THEN
868             GOSUB Background
870             GOSUB Fill_in_values
872             CALL Select_graphics(Which,Last1,Co_ords(*))
874          END IF
876          IF Wave_intrpt THEN GOSUB Wave_menu
878          Interrupted=1
880          ON KEY 9 LABEL "EXIT   PROGRAM",Local_prty+2 GOTO Ret1
882       END LOOP
884 Ret1:  Stp=1 !This will cause the entire program to end.
```

```
886       Interrupted=0
888 Ret2:  OFF KEY  !This will exit only this subprogram.
890       CLEAR SCREEN
892       Intr_prty=Local_prty
894       RETURN
896   !
898   !*****************************************************************
900   !
902 Wave_menu:         !
904       Wave_intrpt=0
906       OFF KEY
908       OFF KBD
910       OFF KNOB
912       Knobcount=0
914       DISP Orange$;"DUT MENU"
916       ON KBD,Local_prty+1 GOSUB Process_kbd
918       ON KNOB .01,Local_prty+1 GOSUB Move_pointer
920       ON KEY 1 LABEL "ACQUIRE  DATA ",Local_prty+1 GOSUB Take_data
922       ON KEY 7 LABEL "MANUAL  SETUP ",Local_prty+1 GOSUB Manual_set
924       ON KEY 5 LABEL "MAIN  MENU ",Local_prty+1 GOTO Ret2
926       SELECT Which
928         !
930       CASE 1 !channel
932         ON KEY 3 LABEL "CHANGE  VALUE",Local_prty+1 GOSUB Sel_chn
934       CASE 2 ! time
936         ON KEY 3 LABEL "CHANGE  VALUE",Local_prty+1 GOSUB Input_time
938         !
940       CASE 3 ! averages
942         ON KEY 3 LABEL "CHANGE  VALUE",Local_prty+1 GOSUB Choose_average
944         !
946       CASE 4 ! voltage
948         ON KEY 3 LABEL "CHANGE  VALUE",Local_prty+1 GOSUB Input_volts
950         !
952       CASE 5     ! delay
954         ON KEY 3 LABEL "CHANGE  VALUE",Local_prty+1 GOSUB Input_delay
956       !
958       CASE 6 ! points
960         ON KEY 3 LABEL "CHANGE  VALUE",Local_prty+1 GOSUB Input_data_pnts
962       !
964       CASE 7 ! offset
966         ON KEY 3 LABEL "CHANGE  VALUE",Local_prty+1 GOSUB Input_offset
968       CASE 8 ! delay reference
970         ON KEY 3 LABEL "CHANGE  VALUE",Local_prty+1 GOSUB Input_ref
972       CASE 9 !trigger level
974         ON KEY 3 LABEL "CHANGE  VALUE",Local_prty+1 GOSUB Input_trig
976       CASE 10!  attenuation
978         ON KEY 3 LABEL "CHANGE  VALUE",Local_prty+1 GOSUB Input_atten
980         !
982       CASE ELSE
984         GOSUB Beeps
986         DISP "ERROR  PARAMETER OUT OF RANGE"
988         WAIT Paused
990       END SELECT
992       !
994       IF Beep_flag THEN BEEP
996       !
```

```
998       RETURN
1000  !
1002  !*************************************************************************
1004  !
1006 Process_kbd:      !
1008      Test$=KBD$
1010      IF LEN(Test$)=1 AND Test$[1,1]<>CHR$(32) THEN RETURN
1012      Wave_intrpt=2
1014      IF Test$[1,1]=CHR$(32) THEN
1016         REPEAT
1018            IF Which<Num_of_choices THEN
1020               Which=Which+1
1022            ELSE
1024               Which=1
1026            END IF
1028         UNTIL Co_ords(Which,1)<>0
1030         CALL Select_graphics(Which,Last1,Co_ords(*))
1032      END IF
1034      IF Test$[1,1]<>CHR$(255) THEN RETURN
1036      SELECT Test$[2,2]
1038      CASE CHR$(255)
1040      ! do nothing, this is a CTRL character
1042      CASE "V","T" !down arrow
1044         REPEAT
1046            IF Which<=(Num_of_choices-Rowsize) THEN
1048               Which=Which+Rowsize
1050            ELSE
1052               Which=Rowsize-(Num_of_choices-Which)
1054            END IF
1056         UNTIL Co_ords(Which,1)<>0
1058      CASE "^","W"     !up arrow
1060         REPEAT
1062            IF Which>Rowsize THEN
1064               Which=Which-Rowsize
1066            ELSE
1068               Which=Which+(Num_of_choices-Rowsize)
1070            END IF
1072         UNTIL Co_ords(Which,1)<>0
1074      CASE "<","H","'"      !left arrow, prev key
1076         REPEAT
1078            IF Which>1 THEN
1080               Which=Which-1
1082            ELSE
1084               Which=Num_of_choices
1086            END IF
1088         UNTIL Co_ords(Which,1)<>0
1090      CASE ">","G",","      !right arrow, next key
1092         REPEAT
1094            IF Which<Num_of_choices THEN
1096               Which=Which+1
1098            ELSE
1100               Which=1
1102            END IF
1104         UNTIL Co_ords(Which,1)<>0
1106      CASE ELSE
1108         BEEP 80.,.1
```

```
1110      END SELECT
1112      CALL Select_graphics(Which,Last1,Co_ords(*))
1114      RETURN
1116 !
1118 !*************************************************************************
1120 !
1122 Move_pointer:    !
1124      Knobcount=Knobcount+KNOBX-KNOBY
1126      IF ABS(Knobcount)<15 THEN RETURN
1128      Wave_intrpt=2
1130      REPEAT
1132         IF Knobcount>0 THEN
1134            Which=Which+1
1136         ELSE
1138            Which=Which-1
1140         END IF
1142         IF Which<1 THEN Which=Num_of_choices
1144         IF Which>Num_of_choices THEN Which=1
1146      UNTIL Co_ords(Which,1)<>0
1148      CALL Select_graphics(Which,Last1,Co_ords(*))
1150      Knobcount=0
1152      RETURN
1154 !
1156 !*************************************************************************
1158 !
1160 Background:      !
1162      !
1164      CLEAR SCREEN
1166      IF Sys_id$[1,4]="S300" THEN
1168         MERGE ALPHA WITH GRAPHICS!This gives back the colors to the alpha
1170                              !plane.
1172      END IF
1174      PRINT TABXY(1,1);Rev_vid$;L_blue$;"   CHANNEL     ";Enh_off$;
1176      PRINT TABXY(1,4);Rev_vid$;L_blue$;" VOLTS/DIV (v) ";Enh_off$;
1178      PRINT TABXY(1,7);Rev_vid$;L_blue$;"  OFFSET (v)   ";Enh_off$;
1180      PRINT TABXY(1,10);Rev_vid$;L_blue$;"  ATTENUATION  ";Enh_off$;
1182      PRINT TABXY(30,1);Rev_vid$;L_blue$;" TIME/DIV (s)  ";Enh_off$;
1184      PRINT TABXY(30,4);Rev_vid$;L_blue$;"   DELAY (s)   ";Enh_off$;
1186      PRINT TABXY(30,7);Rev_vid$;L_blue$;"  DELAY REFER. ";Enh_off$;
1188      PRINT TABXY(59,1);Rev_vid$;L_blue$;"   # OF AVERAGES   ";Enh_off$;
1190      PRINT TABXY(59,4);Rev_vid$;L_blue$;"    # OF POINTS    ";Enh_off$;
1192      PRINT TABXY(59,7);Rev_vid$;L_blue$;" TRIGGER LEVEL (v) ";Enh_off$;
1194      RETURN
1196 !
1198 !*************************************************************************
1200 !
1202 Fill_in_values:      !
1204      GOSUB Print_chnnl
1206      GOSUB Print_time
1208      GOSUB Print_volts
1210      GOSUB Print_pnts
1212      GOSUB Print_ave
1214      GOSUB Print_delay
1216      GOSUB Print_ref
1218      GOSUB Print_offset
1220      GOSUB Print_trig_lev
```

```
1222      GOSUB Print_atten
1224      RETURN
1226 !
1228 !***************************************************************
1230 !
1232      ! = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
1234      ! = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
1236      ! Data input subroutines
1238      ! = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
1240      ! = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
1242 !
1244 !***************************************************************
1246 !
1248 Choose_average:      !
1250      OFF KEY
1252      OFF KNOB
1254      OFF KBD
1256      CLEAR SCREEN
1258      GCLEAR
1260      PRINT L_blue$
1262      Dp$="Select Average "
1264      T$="Available Averages (powers of 2 )    "
1266      Intr_prty=Intr_prty+3
1268      CALL Menu_scroll(Dp$,T$,Allowd$(*),12,1,What(*))
1270      Intr_prty=Intr_prty-3
1272      IF What(1)<>0 THEN ! Aborted
1274        Aver=VAL(Allowd$(What(1)))
1276      END IF
1278      Wave_intrpt=3
1280      !
1282      OUTPUT @Scope;":ACQUIRE:TYPE  AVERAGE"
1284      OUTPUT @Scope;":ACQUIRE:COUNT  "&VAL$(Aver)
1286      RETURN
1288 Print_ave:      !
1290      PRINT TABXY(64,2);Orange$;
1292      CALL Auto_format(Aver*1.0)
1294      RETURN
1296      !
1298 !
1300 !***************************************************************
1302 !
1304 Chnnl_error:  !
1306      BEEP
1308      DISP "Input out of range or disallowed value."
1310      WAIT Paused
1312      DISP "Try again ..."
1314 Sel_chn:      !
1316      ON ERROR GOTO Chnnl_error
1318      Test$=""
1320      INPUT "Enter the acquisition channel number (1-4).",Test$
1322      IF LEN(Test$)<1 THEN RETURN
1324      Temp=VAL(Test$)
1326      OFF ERROR
1328      IF (Temp<1) OR (Temp>4) THEN GOTO Chnnl_error
1330      OUTPUT @Scope;"BLANK CHANNEL"&VAL$(Chnnl)
1332      Chnnl=Temp
```

```
1334      GOSUB New_channl
1336 Print_chnnl:      !
1338      PRINT TABXY(3,2);Orange$;
1340      CALL Auto_format(Chnnl*1.0)
1342      IF NOT (Wave_intrpt) THEN CALL Select_graphics(Which,Last1,Co_ords(*))
1344      RETURN
1346 !
1348 !*****************************************************************
1350 !
1352 New_channl:      !
1354              !This sets the values for the channel just selected
1356              !to the current values.
1358      OUTPUT @Scope;":TRIGGER:LEVEL "&VAL$(Trig)
1360      IF Refer$="    CENTER    " THEN
1362         OUTPUT @Scope;"TIMEBASE:REFERENCE CENTER"
1364      ELSE
1366         OUTPUT @Scope;"TIMEBASE:REFERENCE LEFT"
1368      END IF
1370      OUTPUT @Scope;":TIMEBASE:DELAY "&VAL$(Dly)
1372      OUTPUT @Scope;":TIMEBASE:RANGE "&VAL$(Trange)
1374      OUTPUT @Scope;"VIEW CHANNEL"&VAL$(Chnnl)
1376      OUTPUT @Scope;":CHANNEL"&VAL$(Chnnl)&":OFFSET "&VAL$(Offs)
1378      OUTPUT @Scope;":CHANNEL"&VAL$(Chnnl)&":RANGE "&VAL$(Vrange)
1380      OUTPUT @Scope;":ACQUIRE:TYPE "&Type$
1382      OUTPUT @Scope;":ACQUIRE:COUNT "&VAL$(Aver)
1384      OUTPUT @Scope;":ACQUIRE:POINTS "&VAL$(Pnts)
1386      OUTPUT @Scope;":ACQUIRE:BANDWIDTH HIGH"
1388      RETURN
1390      !
1392 !
1394 !*****************************************************************
1396 !
1398 Input_data_pnts:   !
1400      OFF KEY
1402      OFF KBD
1404      OFF KNOB
1406      CLEAR SCREEN
1408      GCLEAR
1410      PRINT L_blue$
1412      DIM Points$(5)[20]
1414      Dp$="Select Points  "
1416      T$="Available Points for the selected sweep speed."
1418      IF (Time_per_div>=1.E-11) AND (Time_per_div<2.E-11) THEN
1420         Item_cnt=2
1422         REDIM Points$(2)
1424 Frst:    DATA "100"
1426         DATA "400"
1428         RESTORE Frst
1430         READ Points$(*)
1432      END IF
1434      IF (Time_per_div>=2.0E-11) AND (Time_per_div<5.E-11) THEN
1436         Item_cnt=3
1438         REDIM Points$(3)
1440 Secnd:    DATA "100"
1442         DATA "400"
1444         DATA "500"
```

```
1446          RESTORE Secnd
1448          READ Points$(*)
1450        END IF
1452        IF (Time_per_div>=5.0E-11) AND (Time_per_div<2.0E-10) THEN
1454          REDIM Points$(3)
1456          Item_cnt=3
1458 Thrd:     DATA "100"
1460          DATA "500"
1462          DATA "1000"
1464          RESTORE Thrd
1466          READ Points$(*)
1468        END IF
1470        IF (Time_per_div>=2.00E-10) AND (Time_per_div<=1.0) THEN
1472          Item_cnt=5
1474          REDIM Points$(5)
1476 Frth:     DATA "128"
1478          DATA "256"
1480          DATA "500"
1482          DATA "512"
1484          DATA "1024"
1486          RESTORE Frth
1488          READ Points$(*)
1490        END IF
1492        Intr_prty=Intr_prty+3
1494        CALL Menu_scroll(Dp$,T$,Points$(*),Item_cnt,1,What(*))
1496        Intr_prty=Intr_prty-3
1498        IF What(1)<>0 THEN ! Aborted
1500          Pnts=VAL(Points$(What(1)))
1502        END IF
1504        Wave_intrpt=3
1506        !
1508        OUTPUT @Scope;":ACQUIRE:TYPE AVERAGE"
1510        OUTPUT @Scope;":ACQUIRE:COUNT "&VAL$(Pnts)
1512        RETURN
1514 Print_pnts:      !
1516        PRINT TABXY(64,5);Orange$;
1518        CALL Auto_format(Pnts*1.0)
1520        IF NOT (Wave_intrpt) THEN CALL Select_graphics(Which,Last1,Co_ords(*))
1522        RETURN
1524 !
1526 !*************************************************************************
1528 !
1530 Volterr: !
1532        GOSUB Beeps
1534        DISP "ERROR IN READING VALUE OR VALUE OUT OF RANGE, TRY AGAIN."
1536        WAIT Paused
1538 Input_volts:      !
1540        ON ERROR GOSUB Volterr
1542        Test$=""
1544        INPUT "Enter the volts per division in volts ...",Test$
1546        IF LEN(Test$)<1 THEN RETURN
1548        GOSUB Data_check
1550        IF (Rtemp<(Atten*.001)) OR (Rtemp>(Atten*.080)) THEN GOTO Volterr
1552        Volts=Rtemp
1554        Vrange=Volts*8
1556        OUTPUT 707;":CHANNEL"&VAL$(Chnnl)&":RANGE "&VAL$(Vrange)
```

```
1558      OFF ERROR
1560 Print_volts:      !
1562      PRINT TABXY(1,5);Orange$;
1564      CALL Auto_format(Volts)
1566      IF NOT (Wave_intrpt) THEN CALL Select_graphics(Which,Last1,Co_ords(*))
1568      RETURN
1570 !
1572 !*************************************************************************
1574 !
1576 Offset_err: !
1578      GOSUB Beeps
1580      DISP "ERROR IN READING VALUE OR VALUE OUT OF RANGE, TRY AGAIN."
1582      WAIT Paused
1584 Input_offset:      !
1586      ON ERROR GOTO Offset_err
1588      Test$ = ""
1590      INPUT "Enter the offset in volts ...",Test$
1592      IF LEN(Test$) < 1 THEN RETURN
1594      GOSUB Data_check
1596      IF ABS(Rtemp) > 5.00E-1 THEN GOTO Offset_err
1598      Offs = Rtemp
1600      OUTPUT 707;":CHANNEL"&VAL$(Chnnl)&":OFFSET  "&VAL$(Offs)
1602      OFF ERROR
1604 Print_offset:!
1606      PRINT TABXY(1,8);Orange$;
1608      CALL Auto_format(Offs)
1610      IF NOT (Wave_intrpt) THEN CALL Select_graphics(Which,Last1,Co_ords(*))
1612      RETURN
1614 !
1616 !*************************************************************************
1618 !
1620 Atten_err: !
1622      GOSUB Beeps
1624      DISP "ERROR IN READING VALUE OR VALUE OUT OF RANGE, TRY AGAIN."
1626      WAIT Paused
1628 Input_atten:      !
1630      ON ERROR GOTO Atten_err
1632      Test$ = ""
1634      INPUT "Enter the channel attenuation factor...",Test$
1636      IF LEN(Test$) < 1 THEN RETURN
1638      GOSUB Data_check
1640      IF (Rtemp < 1.0) OR (Rtemp > 1000) THEN GOTO Atten_err
1642      Atten = Rtemp
1644      OUTPUT 707;":CHANNEL"&VAL$(Chnnl)&":PROBE  "&VAL$(Atten)
1646      OFF ERROR
1648 Print_atten:      !
1650      PRINT TABXY(3,11);Orange$;
1652      CALL Auto_format(Atten)
1654      IF NOT (Wave_intrpt) THEN CALL Select_graphics(Which,Last1,Co_ords(*))
1656      RETURN
1658 !
1660 !*************************************************************************
1662 !
1664 Time_err:      !
1666      INTEGER Pnterr
1668      !
```

```
1670      Pnterr=0
1672      GOSUB Beeps
1674      DISP "ERROR IN READING VALUE OR VALUE OUT OF RANGE, TRY AGAIN."
1676      WAIT Paused
1678 Input_time:      !
1680      ON ERROR GOSUB Time_err
1682      Test$=""
1684      Temp=Pnts
1686      INPUT "Enter the time per division in seconds ...",Test$
1688      IF LEN(Test$)<1 THEN RETURN
1690      GOSUB Data_check
1692      IF (Rtemp<1.0E-11) OR (Rtemp>1) THEN GOTO Time_err
1694      Time_per_div=Rtemp
1696      Trange=10*Time_per_div
1698      Rtemp=Dly
1700      OUTPUT 707;"TIMEBASE:RANGE "&VAL$(Trange)
1702      OUTPUT 707;":TIMEBASE:DELAY?"
1704      ENTER 707;Dly
1706      IF Rtemp<>Dly THEN
1708         GOSUB Beeps
1710         DISP "The delay value is out of range. Delay set to ",Dly,"."
1712         GOSUB Print_delay
1714         WAIT Paused
1716         DISP Orange$;"DATA MENU "
1718      END IF
1720      OFF ERROR
1722      IF (Time_per_div>=1.E-11) AND (Time_per_div<2.E-11) THEN
1724         SELECT Pnts
1726         CASE 100,400! do nothing these are valid choices.
1728         CASE ELSE
1730            Pnts=400
1732            Pnterr=1
1734         END SELECT
1736      END IF
1738      IF (Time_per_div>=2.0E-11) AND (Time_per_div<5.E-11) THEN
1740         SELECT Pnts
1742         CASE 100,400,500! do nothing, these are valid selections.
1744         CASE ELSE
1746            Pnts=500
1748            Pnterr=1
1750         END SELECT
1752      END IF
1754      IF (Time_per_div>=5.0E-11) AND (Time_per_div<2.0E-10) THEN
1756         SELECT Pnts
1758         CASE 100,500,1000! do nothing, these are valid selections.
1760         CASE ELSE
1762            Pnts=1000
1764            Pnterr=1
1766         END SELECT
1768      END IF
1770      IF (Time_per_div>=2.00E-10) AND (Time_per_div<=1.0) THEN
1772         SELECT Pnts
1774         CASE 128,256,500,512,1024! do nothing, these are valid selections.
1776         CASE ELSE
1778            Pnts=1024
1780            Pnterr=1
```

```
1782        END SELECT
1784      END IF
1786      IF Pnterr THEN
1788        BEEP
1790        DISP "The number of points has been changed to the highest allowed value."
1792        WAIT Paused
1794        DISP "Re-select the number of points if this is not okay."
1796        WAIT Paused
1798        DISP Orange$;"DATA MENU "
1800      END IF
1802      OUTPUT 707;"ACQ:POIN "&VAL$(Pnts)
1804      GOSUB Print_pnts
1806 Print_time:      !
1808      PRINT TABXY(31,2);Orange$;
1810      CALL Auto_format(Time_per_div)
1812      IF NOT (Wave_intrpt) THEN CALL Select_graphics(Which,Last1,Co_ords(*))
1814      RETURN
1816 !
1818 !*************************************************************************
1820 !
1822 Delay_error:!
1824      CLEAR SCREEN
1826      OFF KEY
1828      OFF KNOB
1830      OFF KBD
1832      Wave_intrpt=3
1834      GOSUB Beeps
1836      PRINT "The delay value is not valid for the current"
1838      PRINT "time window. The scope default value is ";Dly;"."
1840 Again: !
1842      PRINT "Your options are to:"
1844      PRINT "A) or <enter>  accept the default value,"
1846      PRINT "B) change the time window to an appropriate value,"
1848      PRINT "C) change the delay."
1850      INPUT "Type the letter corresponding to your desire...",Choice$
1852      IF LEN(Choice$)<1 THEN RETURN
1854      SELECT Choice$
1856      CASE "A","a"
1858        !Do nothing, accept scope value.
1860      CASE "B","b"
1862        Pnts=Temp
1864        GOSUB Input_time
1866        GOSUB Chk_d
1868      CASE "C","c"
1870        GOSUB Input_delay
1872      CASE ELSE
1874        CLEAR SCREEN
1876        GOSUB Beeps
1878        PRINT "That is not an option. Try again"
1880        Choice$=""
1882        GOTO Again
1884      END SELECT
1886      RETURN
1888 Read_delay_err:      !
1890      GOSUB Beeps
1892      DISP "ERROR IN READING DELAY, TRY AGAIN."
```

```
1894      WAIT Paused
1896 Input_delay:  !
1898      ON ERROR GOTO Read_delay_err
1900      Test$ = ""
1902      INPUT "Enter the desired delay time.",Test$
1904      IF LEN(Test$) < 1 THEN RETURN
1906      GOSUB Data_check
1908      OFF ERROR
1910 Chk_d:OUTPUT 707;":TIMEBASE:DELAY "&VAL$(Rtemp)
1912      OUTPUT 707;":TIMEBASE:DELAY?"
1914      ENTER 707;Dly
1916      IF Rtemp < > Dly THEN
1918         GOSUB Delay_error
1920      END IF
1922 Print_delay:      !
1924      PRINT TABXY(31,5);Orange$;
1926      CALL Auto_format(Dly)
1928      IF NOT (Wave_intrpt) THEN CALL Select_graphics(Which,Last1,Co_ords(*))
1930      RETURN
1932 !
1934 !*************************************************************************
1936 !
1938 Input_ref:      !
1940      IF Refer$ = "    CENTER    " THEN
1942         Refer$ = "     LEFT     "
1944         OUTPUT @Scope;"TIMEBASE:REFERENCE LEFT"
1946      ELSE
1948         Refer$ = "    CENTER    "
1950         OUTPUT @Scope;"TIMEBASE:REFERENCE CENTER"
1952      END IF
1954      OUTPUT @Scope;"TIMEBASE:DELAY?"
1956      ENTER @Scope;Dly
1958      GOSUB Print_delay
1960 Print_ref:      !
1962      IF (Refer$ = "left") OR (Refer$ = "LEFT") THEN
1964         Refer$ = "     LEFT     "
1966      END IF
1968      IF (Refer$ = "cent") OR (Refer$ = "CENT") THEN
1970         Refer$ = "    CENTER    "
1972      END IF
1974      PRINT TABXY(30,8);Orange$;
1976      PRINT Refer$
1978      IF NOT (Wave_intrpt) THEN CALL Select_graphics(Which,Last1,Co_ords(*))
1980      RETURN
1982 !
1984 !*************************************************************************
1986 !
1988 Trig_lev_err:              !
1990      GOSUB Beeps
1992      DISP "ERROR IN READING DELAY, TRY AGAIN."
1994      WAIT Paused
1996 Input_trig:      !
1998      ON ERROR GOTO Trig_lev_err
2000      Test$ = ""
2002      INPUT "Enter the desired trigger level in volts.",Test$
2004      IF LEN(Test$) < 1 THEN RETURN
```

```
2006      GOSUB Data_check
2008      IF Rtemp<-1. OR Rtemp>1. THEN
2010         GOSUB Beeps
2012         DISP Red$;"VALUE OUT OF RANGE";Enh_off$;"Try again."
2014         WAIT 1.5
2016         GOTO Input_trig
2018      END IF
2020      Trig=Rtemp
2022      OUTPUT @Scope;":TRIGGER:LEVEL "&VAL$(Trig)
2024      OFF ERROR
2026 Print_trig_lev:              !
2028      PRINT TABXY(60,8);Orange$;
2030      CALL Auto_format(Trig)
2032      IF NOT (Wave_intrpt) THEN CALL Select_graphics(Which,Last1,Co_ords(*))
2034      RETURN
2036 !
2038 !*************************************************************************
2040 !
2042 Manual_set:      !
2044              !this allows the operator to manually set up the scope.
2046      CLEAR SCREEN
2048      OFF KNOB
2050      OFF KBD
2052      OFF KEY
2054      Wave_intrpt=3
2056      DISP "The current channel number is",Chnnl
2058      WAIT Paused
2060      INPUT "Is this the correct channel ? Y/N ",Ch$
2062      IF (Ch$="n") OR (Ch$="N") THEN GOSUB Get_chn
2064      PRINT TABXY(1,1);"Please set up the waveform and     "
2066      PRINT "press continue when done."
2068      LOCAL 707
2070      PAUSE
2072      GOSUB Read_scope_set
2074      RETURN
2076 !
2078 !*************************************************************************
2080 !
2082 Chnl_err:  !
2084      BEEP
2086      DISP "Input out of range or disallowed value."
2088      WAIT Paused
2090      DISP "Try again ..."
2092 Get_chn:     !
2094      ON ERROR GOTO Chnnl_error
2096      Test$=""
2098      INPUT "Enter the acquisition channel number (1-4).",Test$
2100      IF LEN(Test$)<1 THEN RETURN
2102      Temp=VAL(Test$)
2104      OFF ERROR
2106      IF (Temp<1) OR (Temp>4) THEN GOTO Chnnl_error
2108      OUTPUT @Scope;"BLANK CHANNEL"&VAL$(Chnnl)
2110      Chnnl=Temp
2112      OUTPUT @Scope;"VIEW CHANNEL"&VAL$(Chnnl)
2114      OUTPUT @Scope;":ACQUIRE:POINTS "&VAL$(Pnts)
2116      ! The number of point can only be changed from the controller.
```

```
2118      RETURN
2120 !
2122 !*****************************************************************************
2124 !
2126 Read_scp_err:      !
2128      BEEP
2130      DISP "ERROR IN READING SCOPE. WILL TRY Again"
2132 Read_scope_set:      !
2134      ON ERROR GOTO Read_scp_err
2136      OUTPUT @Scope;"ACQUIRE:TYPE?"
2138      ENTER @Scope;Type$
2140      OUTPUT @Scope;"ACQUIRE:COUNT?"
2142      ENTER @Scope;Aver
2144      OUTPUT @Scope;":TIMEBASE:DELAY?"
2146      ENTER @Scope;Dly
2148      OUTPUT @Scope;":TIMEBASE:RANGE?"
2150      ENTER @Scope;Trange
2152      Time_per_div=Trange/10.
2154      OUTPUT @Scope;":TIMEBASE:REFERENCE?"
2156      ENTER @Scope;Refer$
2158      OUTPUT @Scope;":CHANNEL"&VAL$(Chnnl)&":RANGE?"
2160      ENTER @Scope;Vrange
2162      Volts=Vrange/8
2164      OUTPUT @Scope;":CHANNEL"&VAL$(Chnnl)&":OFFSET?"
2166      ENTER @Scope;Offs
2168      OUTPUT @Scope;":CHANNEL"&VAL$(Chnnl)&":PROBE?"
2170      ENTER @Scope;Atten
2172      OUTPUT @Scope;":TRIGGER:LEVEL?"
2174      ENTER @Scope;Trig
2176      OFF ERROR
2178      RETURN
2180 !
2182 !*****************************************************************************
2184 !
2186 Take_data: !
2188      Kill_meas=0
2190      Do_meas=1
2192      OFF KEY
2194      OFF KBD
2196      OFF KNOB
2198      WHILE NOT (Kill_meas) AND (Do_meas)
2200         ON KEY 0 LABEL " ABORT ",Local_prty+5 GOTO Abort_meas
2202         IF Do_meas THEN GOSUB Get_wave
2204      END WHILE
2206      IF NOT (Kill_meas) THEN
2208         OFF KEY
2210         GOSUB Con_to_volts
2212         INPUT "Enter data description, 40 chrs or less. ",Data_id$
2214         Intr_prty=Local_prty+1
2216         CALL Data_to_disk_r(1,INT(Pnts),Voltage(*),Data_id$)
2218         Intr_prty=Local_prty
2220         DEALLOCATE Voltage(*)
2222      END IF
2224      Wave_intrpt=3
2226      RETURN
2228 !
```

```
2230 !*****************************************************************
2232 !
2234 Abort_meas: !
2236     CLEAR @Scope
2238     LOCAL @Scope
2240     Kill_meas=1
2242     CLEAR SCREEN
2244     GOSUB Beeps
2246     DISP Red$;"MEASUREMENT ABORTED";Enh_off$
2248     OFF KEY
2250     OFF KNOB
2252     OFF KBD
2254     OFF ERROR
2256     OFF TIMEOUT 7
2258     CALL Scope_init(Err_flg)
2260     IF Err_flg THEN Interrupted=1
2262     Wave_intrpt=3
2264     RETURN
2266 !
2268 !*****************************************************************
2270 !
2272 Data_error:     !
2274     BEEP
2276     DISP "ERROR IN READING SCOPE. WILL TRY AGAIN"
2278 Get_wave:     !
2280     Do_meas=0
2282     Try_again=1
2284     IF NOT (Kill_meas) THEN
2286        ON ERROR GOTO Data_error
2288        OUTPUT @Scope;"*CLS"
2290        WAIT 1
2292        OUTPUT @Scope;":TER?"
2294        ENTER @Scope;Ter$
2296        OUTPUT @Scope;":NETWORK:REFLECTION:STEP?"
2298        ENTER @Scope;Step$
2300        IF (VAL(Ter$)<>1) AND (TRIM$(Step$)="OFF") THEN
2302           GOSUB Beeps
2304           DISP "No signal detected; please check the setup."
2306           WAIT 2
2308           CLEAR SCREEN
2310           OFF KEY
2312           OFF KNOB
2314           OFF KBD
2316           OFF ERROR
2318           CALL Scope_init(Err_flg)
2320           IF Err_flg THEN Interrupted=1
2322           Wave_intrpt=3
2324           SUBEXIT
2326        END IF
2328        DISP "System busy...."
2330        OUTPUT @Scope;":TIMEBASE:DELAY "&VAL$(Dly)
2332        OUTPUT @Scope;":TIMEBASE:RANGE "&VAL$(Trange)
2334        OUTPUT @Scope;":CHANNEL"&VAL$(Chnnl)&":OFFSET "&VAL$(Offs)
2336        OUTPUT @Scope;":CHANNEL"&VAL$(Chnnl)&":RANGE "&VAL$(Vrange)
2338        OUTPUT @Scope;":ACQUIRE:TYPE AVERAGE"
2340        OUTPUT @Scope;":ACQUIRE:COUNT "&VAL$(Aver)
```

```
2342        OUTPUT @Scope;":ACQUIRE:POINTS "&VAL$(Pnts)
2344        OUTPUT @Scope;"*CLS"
2346        OUTPUT @Scope;"*SRE 32;*ESE 1"
2348        OUTPUT @Scope;":DIGITIZE CHAN"&VAL$(Chnnl)&";*OPC"
2350      ! The following loop allows for a keyboard abort key to be processed
2352      ! before completion of the DIGITIZE command.
2354        WHILE NOT BIT(Stat,5)
2356          Stat=SPOLL(@Scope)
2358        END WHILE
2360        OUTPUT @Scope;":SYSTEM:HEADER OFF;:EOI ON"
2362        OUTPUT @Scope;"WAVEFORM:SOURCE WMEMORY"&VAL$(Chnnl)&"; FORMAT WORD"
2364        OUTPUT @Scope;"WAVEFORM:DATA?"
2366        ENTER @Scope USING "#,A,D";Header$,Bytes
2368        IF Bytes=3 THEN
2370          ENTER @Scope USING "#,3D";Length
2372        END IF
2374        IF Bytes=4 THEN
2376          ENTER @Scope USING "#,4D";Length
2378        END IF
2380        Length=Length/2
2382        IF Pnts<>Length THEN
2384          DISP "The scope will not allow ",Pnts,"points."
2386          WAIT Paused
2388          DISP "The number of points is now ",Length,"."
2390          WAIT Paused
2392          Pnts=Length
2394        END IF
2396        REDIM Waveform(Length)
2398        ENTER @Scope USING "#,W";Waveform(*)
2400        ENTER @Scope USING "-K,B";End$
2402        OUTPUT @Scope;":WAVEFORM:YINCREMENT?"
2404        ENTER @Scope;Yinc
2406        OUTPUT @Scope;":WAVEFORM:YORIGIN?"
2408        ENTER @Scope;Yorg
2410        OUTPUT @Scope;":WAVEFORM:YREFERENCE?"
2412        ENTER @Scope;Yref
2414        OUTPUT @Scope;":WAVEFORM:XINCREMENT?"
2416        ENTER @Scope;Xinc
2418        OUTPUT @Scope;":WAVEFORM:XORIGIN?"
2420        ENTER @Scope;Xorg
2422        OUTPUT @Scope;":WAVEFORM:XREFERENCE?"
2424        ENTER @Scope;Tref
2426        OFF ERROR
2428        IF Bug1 THEN
2430          FOR I=1 TO Pnts
2432            PRINT Waveform(I)
2434          NEXT I
2436        END IF
2438      END IF
2440      RETURN
2442  !
2444  !*************************************************************************
2446  !
2448 Con_to_volts: !
2450      ALLOCATE Voltage(Pnts,2)
2452      FOR I=1 TO Pnts
```

```
2454          Voltage(I,2)=((Waveform(I)-Yref)*Yinc)+Yorg
2456          Voltage(I,1)=((I-1)*Xinc)
2458       NEXT I
2460       RETURN
2462  !
2464  !*************************************************************************
2466  !
2468 Data_check: !
2470       ! The following is a test of the lower case e in a number of
2472       ! scientific notation. If a lower case e occurs, it is converted
2474       ! to upper case. That is all.
2476       IF POS(Test$,"e") THEN
2478          Temp=POS(Test$,"e")
2480          Test$[Temp]="E"&Test$[Temp+1,LEN(Test$)]
2482       END IF
2484       ! end of lower case conversion
2486       Rtemp=VAL(Test$)
2488       RETURN
2490  !
2492  !*************************************************************************
2494  !
2496 Beeps:      !
2498       BEEP 400,.25
2500       BEEP 600,.50
2502       BEEP 400,.25
2504       RETURN
2506       !
2508    SUBEND
2510    !
2512    ! ********************************************************************
2514    !
2516    SUB Vcal
2518 Vcal:  !
2520       CLEAR SCREEN
2522       OPTION BASE 1
2524       DEG
2526       KBD CMODE ON
2528       PRINTER IS CRT
2530       OFF KEY
2532       GOSUB Init
2534       GOSUB Vcal_scope
2536       GOSUB Vcal_cont
2538       SUBEXIT
2540    !
2542    ! ********************************************************************
2544    !
2546 Vcal_scope:  !
2548       ! Sets or re-sets the scope to previously established values.
2550       !
2552       CALL Scope_init(Err_flg)
2554       IF Err_flg THEN
2556          Vcal_intrpt=1
2558          Err_flg=0
2560          SUBEXIT
2562       END IF
2564       OUTPUT @Scope;":ACQUIRE:COUNT "&VAL$(V_aver)
```

```
2566      RETURN
2568  !
2570  !*****************************************************************
2572  !
2574 Init:!
2576      COM /Interrupts/ INTEGER Intr_prty
2578      COM /Sys_msi/ Msi_id$[20]
2580      COM /Sys/ Sys_id$[10]
2582      COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
2584      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
2586      COM /Hue/ Rev_vid$[1],Enh_off$[1],Underline$[1]
2588      COM /Hue/ Red$[1],Orange$[1],L_blue$[1]
2590      COM /Mnu/ INTEGER Interrupted,Which,T_which,V_which,Allowd$(*)
2592      COM /Mnu/ INTEGER Stp
2594      COM /Scope/ REAL Time_per_div,Volts,Trange,Vrange,Dly
2596      COM /Scope/ REAL Probe_fac,Offs,Trig,Atten
2598      COM /Scope/ INTEGER Aver,Pnts,Chnnl
2600      COM /Scope/ Type$[30],Refer$[14],@Scope,Mode$[30]
2602      COM /Vcal_vals/ INTEGER V_aver,Interval,REAL V_step,V_min
2604      COM /Slect/ INTEGER Last1,Tlast,Vlast
2606      !
2608      !
2610      DIM Version$[80]
2612      DIM Test$[160],Data_id$[40]
2614      DIM Choice$[15]
2616      !
2618      !
2620      INTEGER Co_ords(4,4),Lwrlftx,Lwrlfty,Upperrtx,Upperrtty
2622      INTEGER Num_of_choices,What(1)
2624      INTEGER I,J,Pen,Knobcount,Counter
2626      INTEGER Error_flag,Beep_flag,Local_prty,Valid,Datacount
2628      INTEGER Filesize,Baddata,Endpoint,Print_val
2630      INTEGER Yref,Temp,Err_flg
2632      !
2634      !
2636      REAL Data_entered,Data_set_count
2638      REAL Yinc,Yor,Rtemp,Add,Waveform(32767)
2640      Local_prty=Intr_prty
2642      Ftype$="BDAT"
2644      Filesize=500
2646      Last1=0
2648      PEN 1
2650      Beep_flag=0
2652      Paused=1.5
2654      !
2656      !Voltage calibration variables
2658      !
2660      !
2662      RETURN
2664  !
2666  !*****************************************************************
2668  !
2670 Graphs_vcl300:!
2672           !This routine fills the array used by the select_graphics
2674           !routine. The array contains the coordinates for the
2676           !rectangles drawn in select_graphics.
```

```
2678            !if the data is all zeros, this is a null field and is
2680            !used to make all rows of equal length.
2682            !These coordinates are for the system 300 machines.
2684 Frst: DATA 30,900,176,77  !1,1
2686     DATA 665,900,245,77 !1,2
2688     DATA 30,675,176,77 !2,1
2690     DATA 665,675,245,77 !2,2
2692     RESTORE Frst
2694     READ Co_ords(*)
2696     Num_of_choices=4     ! total number of on screen choices
2698     Rowsize=2
2700     RETURN
2702     !
2704 !
2706 !*************************************************************************
2708 !
2710 Vcal_cont:  !
2712     !
2714     OFF KEY
2716     Vcal_intrpt=1
2718     Intr_prty=Local_prty+1
2720     LOOP
2722        IF Vcal_intrpt=1 THEN
2724           IF Sys_id$[1,4]="S300" THEN
2726              CONTROL CRT,5;1
2728              SEPARATE ALPHA FROM GRAPHICS
2730              GOSUB Graphs_vcl300
2732           END IF
2734        END IF
2736        IF (Vcal_intrpt=1) OR (Vcal_intrpt=3) THEN
2738           GOSUB Backgrnd
2740           GOSUB Fill_vals
2742           CALL Select_graphics(V_which,Vlast,Co_ords(*))
2744        END IF
2746        IF Vcal_intrpt THEN GOSUB Vcal_menu
2748        Interrupted=1
2750        ON KEY 9 LABEL "EXIT   PROGRAM",Local_prty+3 GOTO Ret1
2752     END LOOP
2754 Ret1: Stp=1 !This will cause the entire program to end.
2756     Interrupted=0
2758 Ret2: OFF KEY  !This will exit only this subprogram.
2760     CLEAR SCREEN
2762     Intr_prty=Local_prty
2764     RETURN
2766 !
2768 !*************************************************************************
2770 !
2772 Vcal_menu:       !
2774     Vcal_intrpt=0
2776     OFF KEY
2778     OFF KBD
2780     OFF KNOB
2782     Knobcount=0
2784     DISP Orange$;"VCAL MENU "
2786     ON KBD,Local_prty+1 GOSUB Process_kbd
2788     ON KNOB .01,Local_prty+1 GOSUB Move_pointer
```

```
2790     ON KEY 1 LABEL "ACQUIRE  DATA ",Local_prty + 1 GOSUB Vcal_acq
2792     ON KEY 5 LABEL "MAIN  MENU ",Local_prty + 1 GOTO Ret2
2794     ON KEY 7 LABEL "MANUAL SETUP",Local_prty + 1 GOSUB Scp_lcl
2796     SELECT V_which
2798        !
2800     CASE 1 !# of voltage intervals
2802        ON KEY 3 LABEL "CHANGE  VALUE",Local_prty + 1 GOSUB Input_interval
2804        !
2806     CASE 2 ! averages
2808        ON KEY 3 LABEL "CHANGE  VALUE",Local_prty + 1 GOSUB Choose_average
2810        !
2812     CASE 3    !voltage step
2814        ON KEY 3 LABEL "CHANGE  VALUE",Local_prty + 1 GOSUB Input_step
2816     !
2818     CASE 4 ! minimum calibration voltage
2820        ON KEY 3 LABEL "CHANGE  VALUE",Local_prty + 1 GOSUB Input_min_volt
2822        !
2824        !
2826     CASE ELSE
2828        GOSUB Beeps
2830        DISP "ERROR  PARAMETER OUT OF RANGE"
2832        WAIT Paused
2834     END SELECT
2836     !
2838     RETURN
2840 !
2842 !*************************************************************************
2844 !
2846 Process_kbd:     !
2848     Test$=KBD$
2850     IF LEN(Test$)=1 AND Test$[1,1]<>CHR$(32) THEN RETURN
2852     Vcal_intrpt=2
2854     IF Test$[1,1]=CHR$(32) THEN
2856        REPEAT
2858           IF V_which<Num_of_choices THEN
2860              V_which=V_which+1
2862           ELSE
2864              V_which=1
2866           END IF
2868        UNTIL Co_ords(V_which,1)<>0
2870        CALL Select_graphics(V_which,Vlast,Co_ords(*))
2872     END IF
2874     IF Test$[1,1]<>CHR$(255) THEN RETURN
2876     SELECT Test$[2,2]
2878     CASE CHR$(255)
2880     ! do nothing, this is a CTRL character
2882     CASE "V","T" !down arrow
2884        REPEAT
2886           IF V_which<=(Num_of_choices-Rowsize) THEN
2888              V_which=V_which+Rowsize
2890           ELSE
2892              V_which=Rowsize-(Num_of_choices-V_which)
2894           END IF
2896        UNTIL Co_ords(V_which,1)<>0
2898     CASE "^","W"     !up arrow
2900        REPEAT
```

```
2902            IF V_which>Rowsize THEN
2904               V_which=V_which-Rowsize
2906            ELSE
2908               V_which=V_which+(Num_of_choices-Rowsize)
2910            END IF
2912         UNTIL Co_ords(V_which,1)<>0
2914      CASE "<","H","'"     !left arrow, prev key
2916         REPEAT
2918            IF V_which>1 THEN
2920               V_which=V_which-1
2922            ELSE
2924               V_which=Num_of_choices
2926            END IF
2928         UNTIL Co_ords(V_which,1)<>0
2930      CASE ">","G","."     !right arrow, next key
2932         REPEAT
2934            IF V_which<Num_of_choices THEN
2936               V_which=V_which+1
2938            ELSE
2940               V_which=1
2942            END IF
2944         UNTIL Co_ords(V_which,1)<>0
2946      CASE ELSE
2948         BEEP 80.,.1
2950      END SELECT
2952      CALL Select_graphics(V_which,Vlast,Co_ords(*))
2954      RETURN
2956 !
2958 !*************************************************************************
2960 !
2962 Move_pointer:    !
2964      Knobcount=Knobcount+KNOBX-KNOBY
2966      IF ABS(Knobcount)<15 THEN RETURN
2968      Vcal_intrpt=2
2970      REPEAT
2972         IF Knobcount>0 THEN
2974            V_which=V_which+1
2976         ELSE
2978            V_which=V_which-1
2980         END IF
2982         IF V_which<1 THEN V_which=Num_of_choices
2984         IF V_which>Num_of_choices THEN V_which=1
2986      UNTIL Co_ords(V_which,1)<>0
2988      CALL Select_graphics(V_which,Vlast,Co_ords(*))
2990      Knobcount=0
2992      RETURN
2994 !
2996 !*************************************************************************
2998 !
3000 Backgrnd:      !
3002      !
3004      CLEAR SCREEN
3006      IF Sys_id$[1,4]="S300" THEN
3008         MERGE ALPHA WITH GRAPHICS!This gives back the colors to the alpha
3010                              !plane.
3012      END IF
```

```
3014      PRINT TABXY(1,1);Rev_vid$;L_blue$;" # OF VOLTAGE  ";Enh_off$;
3016      PRINT TABXY(1,2);Rev_vid$;L_blue$;"   INTERVALS   ";Enh_off$
3018      PRINT TABXY(1,7);Rev_vid$;L_blue$;" VOLTAGE  STEP ";Enh_off$;
3020      PRINT TABXY(1,8);Rev_vid$;L_blue$;"  SIZE (VOLTS) ";Enh_off$;
3022      PRINT TABXY(55,1);Rev_vid$;L_blue$;"       NUMBER OF     ";Enh_off$;
3024      PRINT TABXY(55,2);Rev_vid$;L_blue$;"       AVERAGES      ";Enh_off$;
3026      PRINT TABXY(55,7);Rev_vid$;L_blue$;" MINIMUM CALIBRATION ";Enh_off$;
3028      PRINT TABXY(55,8);Rev_vid$;L_blue$;"   VOLTAGE (VOLTS)   ";Enh_off$;
3030      RETURN
3032 !
3034 !*************************************************************************
3036 !
3038 Fill_vals:      !
3040      GOSUB Print_ave
3042      GOSUB Print_intervls
3044      GOSUB Print_step
3046      GOSUB Print_min_cal
3048      RETURN
3050 !
3052 !*************************************************************************
3054 !
3056 Scp_lcl:      !
3058              !this allows the operator to manually set up the scope.
3060              !
3062      OFF KBD
3064      OFF KEY
3066      OFF KNOB
3068      Vcal_intrpt=3
3070      CLEAR SCREEN
3072      PRINT TABXY(1,1);" This is to establish the number of voltage "
3074      PRINT " intervals needed and the minimum voltage level required "
3076      PRINT " for calibrating the voltage levels of the oscilloscope. "
3078      PRINT " Any changes made to the scope settings will be re-set to "
3080      PRINT " the values which were present before this option was "
3082      PRINT " invoked. If changes are required, please use either the "
3084      PRINT " waveform acquisition menu or the voltage calibration menu. "
3086      PRINT " Voltage calibration data is not valid for a DUT waveform "
3088      PRINT " which has a different sensitivity and/or offset."
3090      !
3092      PRINT TABXY(1,21);"The current channel is ",Chnnl
3094      INPUT "is this the correct channel ? Y/N ",Ch$
3096      IF (Ch$="n") OR (Ch$="N") THEN GOSUB Get_chn
3098      CLEAR SCREEN
3100      !
3102      PRINT TABXY(1,24);"Please set up the waveform and     "
3104      PRINT "press continue when done."
3106      LOCAL 707
3108      CALL Pause_key_on
3110      OFF KEY
3112      GOSUB Vcal_scope
3114      RETURN
3116 !
3118 !*************************************************************************
3120 !
3122 Chnl_err:  !
3124      BEEP
```

```
3126      DISP "Input out of range or disallowed value."
3128      WAIT Paused
3130      DISP "Try again ..."
3132 Get_chn:     !
3134      ON ERROR GOTO Chnl_err
3136      Test$ = ""
3138      INPUT "Enter the acquisition channel number (1-4).",Test$
3140      IF LEN(Test$) < 1 THEN RETURN
3142      Temp = VAL(Test$)
3144      OFF ERROR
3146      IF (Temp < 1) OR (Temp > 4) THEN GOTO Chnnl_error
3148      OUTPUT @Scope;"BLANK CHANNEL"&VAL$(Chnnl)
3150      Chnnl = Temp
3152      OUTPUT @Scope;"VIEW CHANNEL"&VAL$(Chnnl)
3154      ! The number of point can only be changed from the controller.
3156      RETURN
3158 !
3160 !*************************************************************************
3162 !
3164      !
3166      ! = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
3168      ! = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
3170      ! Data input subroutines
3172      ! = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
3174      ! = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
3176 !
3178 !*************************************************************************
3180 !
3182 Choose_average:      !
3184      OFF KEY
3186      OFF KNOB
3188      OFF KBD
3190      CLEAR SCREEN
3192      GCLEAR
3194      PRINT L_blue$
3196      DIM Dp$[80]
3198      DIM T$[52]
3200      Dp$ = "Select Average "
3202      T$ = "Available Averages (powers of 2 )    "
3204      Intr_prty = Intr_prty + 3
3206      CALL Menu_scroll(Dp$,T$,Allowd$(*),12,1,What(*))
3208      Intr_prty = Intr_prty-3
3210      IF What(1) < > 0 THEN ! Aborted
3212         V_aver = VAL(Allowd$(What(1)))
3214      END IF
3216      Vcal_intrpt = 3
3218      !
3220      OUTPUT @Scope;":ACQUIRE:TYPE AVERAGE"
3222      OUTPUT @Scope;":ACQUIRE:COUNT "&VAL$(V_aver)
3224      RETURN
3226 Print_ave:      !
3228      PRINT TABXY(60,4);Orange$;
3230      CALL Auto_format(V_aver*1.0)
3232      RETURN
3234      !
3236 !
```

```
3238 !*************************************************************************
3240 !
3242 Interval_error: !
3244      GOSUB Beeps
3246      DISP "ERROR IN READING VALUE OR DISALLOWED VALUE, TRY AGAIN."
3248      WAIT Paused
3250 Input_interval: !
3252      ON ERROR GOTO Interval_error
3254      Test$ = ""
3256      INPUT "Enter the number of voltage intervals.",Test$
3258      IF LEN(Test$) < 1 THEN RETURN
3260      Temp = VAL(Test$)
3262      OFF ERROR
3264      IF Temp < 1 THEN GOTO Interval_error
3266      Interval = Temp
3268 Print_intervls:     !
3270      PRINT TABXY(2,4);Orange$;
3272      CALL Auto_format(Interval * 1.0)
3274      IF NOT (Vcal_intrpt) THEN CALL Select_graphics(V_which,Vlast,Co_ords(*))
3276      RETURN
3278 !
3280 !*************************************************************************
3282 !
3284 Step_error: !
3286      GOSUB Beeps
3288      DISP "ERROR IN READING VALUE OR VALUE OUT OF RANGE. TRY AGAIN."
3290      WAIT Paused
3292 Input_step:     !
3294      ON ERROR GOTO Step_error
3296      Test$ = ""
3298      DISP "Enter the voltage step size."
3300      INPUT "This must be a positive, real number.",Test$
3302      IF LEN(Test$) < 1 THEN RETURN
3304      GOSUB Data_check
3306      OFF ERROR
3308      IF Rtemp < 0. THEN GOTO Step_error
3310      V_step = Rtemp
3312 Print_step:     !
3314      PRINT TABXY(1,10);Orange$;
3316      CALL Auto_format(V_step)
3318      IF NOT (Vcal_intrpt) THEN CALL Select_graphics(V_which,Vlast,Co_ords(*))
3320      RETURN
3322 !
3324 !*************************************************************************
3326 !
3328 Minvolt_error: !
3330      GOSUB Beeps
3332      DISP "ERROR IN READING VALUE OR VALUE OUT OF RANGE. TRY AGAIN."
3334      WAIT Paused
3336 Input_min_volt:     !
3338      ON ERROR GOTO Minvolt_error
3340      Test$ = ""
3342      INPUT "Enter the minimum calibration voltage.",Test$
3344      IF LEN(Test$) < 1 THEN RETURN
3346      GOSUB Data_check
3348      OFF ERROR
```

```
3350      V_min=Rtemp
3352 Print_min_cal:      !
3354      PRINT TABXY(59,10);Orange$;
3356      CALL Auto_format(V_min)
3358      IF NOT (Vcal_intrpt) THEN CALL Select_graphics(V_which,Vlast,Co_ords(*))
3360      RETURN
3362 !
3364 !*****************************************************************
3366 !
3368 Vcal_acq: !
3370      ALLOCATE Vcal_ary((Interval+2),2)
3372      ! Vcal_ary(1,1) = # of voltage levels measured.
3374      ! Vcal_ary(i,1) = the measured voltage. i=2 to interval+2
3376      ! Vcal_ary(i,2) = the calibration voltage.
3378      Counter=2
3380      V_true=V_min
3382      Kill_meas=0
3384      Vcal_ary(1,1)=Interval+1
3386      OFF KEY
3388      OFF KNOB
3390      OFF KBD
3392      WHILE NOT (Kill_meas) AND (Counter<=(Interval+2))
3394         ON KEY 0 LABEL " ABORT ",Local_prty+5 GOTO Abort_vmeas
3396         BEEP
3398         DISP "Set input voltage to";V_true;" press continue when ready."
3400         PAUSE
3402         GOSUB Get_vwave
3404         IF NOT (Kill_meas) THEN
3406            GOSUB Con_to_volts
3408            Add=0
3410            FOR J=1 TO Pnts
3412               Add=Add+Voltage(J,2)
3414            NEXT J
3416            Add=Add/Pnts
3418            !add is the average dc voltage measured.
3420            Vcal_ary(Counter,1)=Add
3422            Vcal_ary(Counter,2)=V_true
3424            V_true=V_true+V_step
3426            Counter=Counter+1
3428            DEALLOCATE Voltage(*)
3430         END IF
3432      END WHILE
3434      IF NOT (Kill_meas) THEN
3436         OFF KEY
3438         CLEAR SCREEN
3440         PRINT TABXY(1,1);"There were ";Vcal_ary(1,1);" levels measured."
3442         PRINT TABXY(1,3);"MEASURED VOLTAGE";
3444         PRINT TABXY(30,3);"CALIBRATION VOLTAGE";
3446         FOR I=2 TO Vcal_ary(1,1)+1
3448            PRINT TABXY(2,I+3);Vcal_ary(I,1);
3450            PRINT TABXY(32,I+3);Vcal_ary(I,2);
3452         NEXT I
3454         INPUT "Is the data Okay? y/n",Ch$
3456         IF Ch$="n" OR Ch$="N" THEN
3458            DISP "Please re-set the measurement and try again."
3460            WAIT 1.5
```

```
3462          ELSE
3464             INPUT "Enter data description, 40 chrs or less. ",Data_id$
3466             Intr_prty=Local_prty+1
3468             CALL Data_to_disk_r(1,INT((Counter+1)),Vcal_ary(*),Data_id$)
3470             Intr_prty=Local_prty
3472          END IF
3474       END IF
3476       DEALLOCATE Vcal_ary(*)
3478       OFF KEY
3480       Vcal_intrpt=3
3482       RETURN
3484  !
3486  !*************************************************************************
3488  !
3490 Abort_vmeas: !
3492       CLEAR @Scope
3494       LOCAL @Scope
3496       Kill_meas=1
3498       CLEAR SCREEN
3500       DISP Red$;"MEASUREMENT ABORTED";Enh_off$
3502       OFF KEY
3504       OFF KNOB
3506       OFF KBD
3508       OFF ERROR
3510       Vcal_intrpt=3
3512       GOTO Vcal_scope
3514  !
3516  !*************************************************************************
3518 Data_error:      !
3520       BEEP
3522       DISP "ERROR IN READING SCOPE. WILL TRY AGAIN"
3524       WAIT Paused
3526 Get_vwave:      !
3528       Do_meas=0
3530       IF NOT (Kill_meas) THEN
3532          ON ERROR GOTO Data_error
3534          OUTPUT @Scope;"*CLS"
3536          WAIT 1
3538          OUTPUT @Scope;":TER?"
3540          ENTER @Scope;Ter$
3542          IF (VAL(Ter$)<>1) THEN
3544             GOSUB Beeps
3546             DISP "No signal detected; please check the setup."
3548             WAIT 2
3550             CLEAR SCREEN
3552             OFF KEY
3554             OFF KNOB
3556             OFF KBD
3558             OFF ERROR
3560             CALL Scope_init(Err_flg)
3562             IF Err_flg THEN Interrupted=1
3564             Vcal_intrpt=3
3566             SUBEXIT
3568          END IF
3570          DISP "System busy; Measurement Number";Counter-1;"of";Interval+1
3572          OUTPUT @Scope;":TIMEBASE:DELAY "&VAL$(Dly)
```

```
3574        OUTPUT @Scope;":TIMEBASE:RANGE "&VAL$(Trange)
3576        OUTPUT @Scope;":CHANNEL"&VAL$(Chnnl)&":OFFSET "&VAL$(Offs)
3578        OUTPUT @Scope;":CHANNEL"&VAL$(Chnnl)&":RANGE "&VAL$(Vrange)
3580        OUTPUT @Scope;":ACQUIRE:TYPE AVERAGE"
3582        OUTPUT @Scope;":ACQUIRE:COUNT "&VAL$(V_aver)
3584        OUTPUT @Scope;":ACQUIRE:POINTS "&VAL$(Pnts)
3586        OUTPUT @Scope;"*CLS"
3588        OUTPUT @Scope;"*SRE 32;*ESE 1"
3590        OUTPUT @Scope;":DIGITIZE CHAN"&VAL$(Chnnl)&";*OPC"
3592        ! Digitizes the channel and asks for the operation complete
3594        ! status indicator flag.
3596        !
3598        ! This loop enables the keyboard interrupt key called ABORT.
3600        ! The abort key will work even when the scope is busy.
3602        !
3604        WHILE NOT BIT(Stat,5)
3606           Stat=SPOLL(@Scope)
3608        END WHILE
3610        OUTPUT @Scope;":SYSTEM:HEADER OFF;:EOI ON"
3612        OUTPUT @Scope;"WAVEFORM:SOURCE WMEMORY"&VAL$(Chnnl)&"; FORMAT WORD"
3614        OUTPUT @Scope;"WAVEFORM:DATA?"
3616        ENTER @Scope USING "#,A,D";Header$,Bytes
3618        IF Bytes=3 THEN
3620           ENTER @Scope USING "#,3D";Length
3622        END IF
3624        IF Bytes=4 THEN
3626           ENTER @Scope USING "#,4D";Length
3628        END IF
3630        Length=Length/2
3632        IF Pnts<>Length THEN
3634           DISP "The scope will not allow ",Pnts,"points."
3636           WAIT Paused
3638           DISP "The number of points is now ",Length,"."
3640           WAIT Paused
3642           Pnts=Length
3644        END IF
3646        REDIM Waveform(Length)
3648        ENTER @Scope USING "#,W";Waveform(*)
3650        ENTER @Scope USING "-K,B";End$
3652        OUTPUT @Scope;":WAVEFORM:YINCREMENT?"
3654        ENTER @Scope;Yinc
3656        OUTPUT @Scope;":WAVEFORM:YORIGIN?"
3658        ENTER @Scope;Yorg
3660        OUTPUT @Scope;":WAVEFORM:YREFERENCE?"
3662        ENTER @Scope;Yref
3664        OUTPUT @Scope;":WAVEFORM:XINCREMENT?"
3666        ENTER @Scope;Xinc
3668        OUTPUT @Scope;":WAVEFORM:XORIGIN?"
3670        ENTER @Scope;Xorg
3672        OUTPUT @Scope;":WAVEFORM:XREFERENCE?"
3674        ENTER @Scope;Tref
3676        OFF ERROR
3678        IF Bug1 THEN
3680           FOR I=1 TO Pnts
3682              PRINT Waveform(I)
3684           NEXT I
```

```
3686          END IF
3688       END IF
3690       RETURN
3692  !
3694  !*************************************************************************
3696  !
3698 Con_to_volts: !
3700       ALLOCATE Voltage(Pnts,2)
3702       FOR I=1 TO Pnts
3704          Voltage(I,2)=((Waveform(I)-Yref)*Yinc)+Yorg
3706          Voltage(I,1)=((I-1)*Xinc)
3708       NEXT I
3710       RETURN
3712  !
3714  !*************************************************************************
3716  !
3718 Data_check: !
3720       ! The following is a test of the lower case e in a number of
3722       ! scientific notation. If a lower case e occurs, it is converted
3724       ! to upper case. That is all.
3726       IF POS(Test$,"e") THEN
3728          Temp=POS(Test$,"e")
3730          Test$[Temp]="E"&Test$[Temp+1,LEN(Test$)]
3732       END IF
3734       ! end of lower case conversion
3736       Rtemp=VAL(Test$)
3738       RETURN
3740  !
3742  !*************************************************************************
3744  !
3746 Beeps:      !
3748       BEEP 400,.25
3750       BEEP 600,.50
3752       BEEP 400,.25
3754       RETURN
3756       !
3758    SUBEND
3760    !
3762    ! ********************************************************************
3764    !
3766    SUB Tcal
3768 Tcal:  !
3770       CLEAR SCREEN
3772       OPTION BASE 1
3774       KBD CMODE ON
3776       PRINTER IS CRT
3778       OFF KEY
3780       GOSUB Tcal_vars
3782       GOSUB Tcal_scope
3784       GOSUB Tcal_cont
3786       SUBEXIT
3788    !
3790    ! ********************************************************************
3792    !
3794 Tcal_vars:!
3796       COM /Interrupts/ INTEGER Intr_prty
```

```
3798      COM /Sys_msi/ Msi_id$[20]
3800      COM /Sys/ Sys_id$[10]
3802      COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
3804      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
3806      COM /Hue/ Rev_vid$[1],Enh_off$[1],Underline$[1]
3808      COM /Hue/ Red$[1],Orange$[1],L_blue$[1]
3810      COM /Mnu/ INTEGER Interrupted,Which,T_which,V_which,Allowd$(*)
3812      COM /Mnu/ INTEGER Stp
3814      COM /Scope/ REAL Time_per_div,Volts,Trange,Vrange,Dly
3816      COM /Scope/ REAL Probe_fac,Offs,Trig,Atten
3818      COM /Scope/ INTEGER Aver,Pnts,Chnnl
3820      COM /Scope/ Type$[30],Refer$[14],@Scope,Mode$[30]
3822      COM /Tcal_vals/ INTEGER Zero_x,Ls_prs,Slope,T_aver,REAL Freq
3824      COM /Tcal_vals/ Slpe$[10],Save$[10],REAL Tc_off,Tc_volt,T_trig
3826      COM /Line_fit/ !NTEGER Wav(1024)
3828      COM /Slect/ INTEGER Last1,Tlast,Vlast
3830      !
3832      !
3834      DIM Version$[80]
3836      DIM Test$[160],Data_id$[40]
3838      DIM Choice$[15]
3840      !
3842      !
3844      INTEGER Co_ords(9,4),Lwrlftx,Lwrlfty,Upperrtx,Upperrtty
3846      INTEGER Num_of_choices,What(1)
3848      INTEGER !,Cntr,Pen,Knobcount
3850      INTEGER Error_flag,Beep_flag,Local_prty,Valid,Datacount
3852      !NTEGER Filesize,Baddata,Endpoint,Print_val
3854      INTEGER Yref,Temp,Tcal_intrpt,Kount
3856      INTEGER Strt,End_pnt,Midval,Err_flg
3858      !
3860      !
3862      REAL Data_entered,Data_set_count
3864      REAL Yinc,Yor,Rtemp,Zero_level,Ln_slp,Intrcpt
3866      REAL Maxval,Minval,Waveform(32767)
3868      !
3870      !
3872      Local_prty=Intr_prty
3874      Ftype$="BDAT"
3876      Filesize=32767
3878      Last1=0
3880      PEN 1
3882      Beep_flag=0
3884      Paused=1.5
3886      !
3888      !Time calibration variables
3890      !
3892      !
3894      RETURN
3896 !
3898 !*************************************************************************
3900 !
3902 Tcal_scope:     !
3904      CALL Scope_init(Err_flg)
3906      IF Err_flg THEN
3908         Tcal_intrpt=3
```

```
3910        Err_flg=0
3912        SUBEXIT
3914      END IF
3916      OUTPUT @Scope;":CHANNEL"&VAL$(Chnnl)&":OFFSET  "&VAL$(Tc_off)
3918      Tc_vrange=Tc_volt*8
3920      OUTPUT @Scope;":CHANNEL"&VAL$(Chnnl)&":RANGE  "&VAL$(Tc_vrange)
3922      OUTPUT @Scope;":ACQUIRE:COUNT  "&VAL$(T_aver)
3924      OUTPUT @Scope;"TRIGGER:LEVEL  "&VAL$(T_trig)
3926      RETURN
3928      !
3930 !
3932 !*************************************************************************
3934 !
3936 !
3938 Graphs_ary300:!
3940            !This routine fills the array used by the select_graphics
3942            !routine. The array contains the coordinates for the
3944            !rectangles drawn in select_graphics.
3946            !if the data is all zeros, this is a null field and is
3948            !used to make all rows of equal length.
3950            !These coordinates are for the system 300 machines.
3952 First:DATA 30,943,176,34  !1,1 frequency
3954      DATA 372,943,245,34 !1,2 save the waveform
3956      DATA 714,943,220,34 !1,3 averages
3958      DATA 30,830,176,35 !2,1 volts
3960      DATA 372,830,245,35 !2,2 calibrate on (pos/neg slope)
3962      DATA 714,793,220,70 !2,3 number of points for the moving average
3964      DATA 30,717,176,35  !3,1 offset mv
3966      DATA 372,680,245,70 !3,2 number of least squares point pairs
3968      DATA 714,680,220,35 !3,3 trigger level for time calibration
3970      RESTORE First
3972      READ Co_ords(*)
3974      Num_of_choices=9    ! total number of on screen choices
3976      Rowsize=3
3978      RETURN
3980      !
3982 !
3984 !*************************************************************************
3986 !
3988 Tcal_cont:  !
3990      !
3992      OFF KEY
3994      Tcal_intrpt=1
3996      LOOP
3998         IF Tcal_intrpt=1 THEN
4000            IF Sys_id$[1,4]="S300" THEN
4002               CONTROL CRT,5;1
4004               SEPARATE ALPHA FROM GRAPHICS
4006               GOSUB Graphs_ary300
4008            END IF
4010         END IF
4012         IF (Tcal_intrpt=1) OR (Tcal_intrpt=3) THEN
4014            GOSUB Background
4016            GOSUB Fill_in
4018            CALL Select_graphics(T_which,Tlast,Co_ords(*))
4020         END IF
```

```
4022        IF Tcal_intrpt THEN GOSUB Tcal_menu
4024        Interrupted=1
4026        ON KEY 9 LABEL "EXIT   PROGRAM",Local_prty+2 GOTO Ret1
4028      END LOOP
4030 Ret1: Stp=1 !This will cause the entire program to end.
4032 Ret2: OFF KEY  !This will exit only this subprogram.
4034      CLEAR SCREEN
4036      RETURN
4038 !
4040 !*************************************************************************
4042 !
4044 Tcal_menu:       !
4046      Tcal_intrpt=0
4048      OFF KEY
4050      OFF KBD
4052      OFF KNOB
4054      Knobcount=0
4056      DISP Orange$;"TCAL MENU "
4058      ON KBD,Local_prty+1 GOSUB Process_kbd
4060      ON KNOB .01,Local_prty+1 GOSUB Move_pointer
4062      ON KEY 1 LABEL "ACQUIRE  DATA ",Local_prty+1 GOSUB Tcal_data
4064      ON KEY 5 LABEL "MAIN  MENU ",Local_prty+1 GOTO Ret2
4066      ON KEY 7 LABEL "MANUAL SETUP",Local_prty+1 GOSUB Tcal_set
4068      SELECT T_which
4070        !
4072      CASE 1 !frequency
4074        ON KEY 3 LABEL "CHANGE  VALUE",Local_prty+1 GOSUB Input_freq
4076      CASE 2 !save the sinewave?
4078        ON KEY 3 LABEL "CHANGE  VALUE",Local_prty+1 GOSUB Save_wave
4080        !
4082      CASE 3 ! # of point pairs
4084        ON KEY 3 LABEL "CHANGE  VALUE",Local_prty+1 GOSUB Choose_average
4086        !
4088      CASE 4 ! voltage
4090        ON KEY 3 LABEL "CHANGE  VALUE",Local_prty+1 GOSUB Input_volts
4092        !
4094      CASE 5     ! slope
4096        ON KEY 3 LABEL "CHANGE  VALUE",Local_prty+1 GOSUB Which_slope
4098      !
4100      CASE 6 ! points for moving average
4102        ON KEY 3 LABEL "CHANGE  VALUE",Local_prty+1 GOSUB Moving_points
4104      !
4106      CASE 7 ! offset
4108        ON KEY 3 LABEL "CHANGE  VALUE",Local_prty+1 GOSUB Input_offset
4110        !
4112      CASE 8     ! least squares point pairs
4114        ON KEY 3 LABEL "CHANGE  VALUE",Local_prty+1 GOSUB Point_pairs
4116        !
4118      CASE 9     ! trigger level
4120        ON KEY 3 LABEL "CHANGE  VALUE",Local_prty+1 GOSUB Trigger_lvl
4122        !
4124      CASE ELSE
4126        GOSUB Beeps
4128        DISP "ERROR  PARAMETER OUT OF RANGE"
4130        WAIT Paused
4132      END SELECT
```

```
4134      !
4136      RETURN
4138  !
4140  !*************************************************************************
4142  !
4144 Process_kbd:      !
4146      Test$=KBD$
4148      IF LEN(Test$)=1 AND Test$[1,1]<>CHR$(32) THEN RETURN
4150      Tcal_intrpt=2
4152      IF Test$[1,1]=CHR$(32) THEN
4154         REPEAT
4156            IF T_which<Num_of_choices THEN
4158               T_which=T_which+1
4160            ELSE
4162               T_which=1
4164            END IF
4166         UNTIL Co_ords(T_which,1)<>0
4168         CALL Select_graphics(T_which,Tlast,Co_ords(*))
4170      END IF
4172      IF Test$[1,1]<>CHR$(255) THEN RETURN
4174      SELECT Test$[2,2]
4176      CASE CHR$(255)
4178      ! do nothing, this is a CTRL character
4180      CASE "V","T" !down arrow
4182         REPEAT
4184            IF T_which<=(Num_of_choices-Rowsize) THEN
4186               T_which=T_which+Rowsize
4188            ELSE
4190               T_which=Rowsize-(Num_of_choices-T_which)
4192            END IF
4194         UNTIL Co_ords(T_which,1)<>0
4196      CASE "^","W"     !up arrow
4198         REPEAT
4200            IF T_which>Rowsize THEN
4202               T_which=T_which-Rowsize
4204            ELSE
4206               T_which=T_which+(Num_of_choices-Rowsize)
4208            END IF
4210         UNTIL Co_ords(T_which,1)<>0
4212      CASE "<","H","'"     !left arrow, prev key
4214         REPEAT
4216            IF T_which>1 THEN
4218               T_which=T_which-1
4220            ELSE
4222               T_which=Num_of_choices
4224            END IF
4226         UNTIL Co_ords(T_which,1)<>0
4228      CASE ">","G","."     !right arrow, next key
4230         REPEAT
4232            IF T_which<Num_of_choices THEN
4234               T_which=T_which+1
4236            ELSE
4238               T_which=1
4240            END IF
4242         UNTIL Co_ords(T_which,1)<>0
4244      CASE ELSE
```

```
4246        BEEP 80.,.1
4248      END SELECT
4250      CALL Select_graphics(T_which,Tlast,Co_ords(*))
4252      RETURN
4254 !
4256 !*************************************************************************
4258 !
4260 Move_pointer:    !
4262      Knobcount=Knobcount+KNOBX-KNOBY
4264      IF ABS(Knobcount)<15 THEN RETURN
4266      Tcal_intrpt=2
4268      REPEAT
4270        IF Knobcount>0 THEN
4272          T_which=T_which+1
4274        ELSE
4276          T_which=T_which-1
4278        END IF
4280        IF T_which<1 THEN T_which=Num_of_choices
4282        IF T_which>Num_of_choices THEN T_which=1
4284      UNTIL Co_ords(T_which,1)<>0
4286      CALL Select_graphics(T_which,Tlast,Co_ords(*))
4288      Knobcount=0
4290      RETURN
4292 !
4294 !*************************************************************************
4296 !
4298 Background:      !
4300      !
4302      CLEAR SCREEN
4304      IF Sys_id$[1,4]="S300" THEN
4306        MERGE ALPHA WITH GRAPHICS!This gives back the colors to the alpha
4308                        !plane.
4310      END IF
4312      PRINT TABXY(1,1);Rev_vid$;L_blue$;"   FREQUENCY   ";Enh_off$;
4314      PRINT TABXY(1,4);Rev_vid$;L_blue$;" VOLTS/DIV (v) ";Enh_off$;
4316      PRINT TABXY(1,7);Rev_vid$;L_blue$;"  OFFSET (v)   ";Enh_off$;
4318      PRINT TABXY(30,1);Rev_vid$;L_blue$;" SAVE THE WAVEFORM ? ";Enh_off$;
4320      PRINT TABXY(30,4);Rev_vid$;L_blue$;"    CALIBRATE ON    ";Enh_off$;
4322      PRINT TABXY(30,7);Rev_vid$;L_blue$;"  POINT PAIRS FOR   ";Enh_off$;
4324      PRINT TABXY(30,8);Rev_vid$;L_blue$;" LEAST SQUARES FIT  ";Enh_off$;
4326      PRINT TABXY(59,1);Rev_vid$;L_blue$;"  # OF AVERAGES    ";Enh_off$;
4328      PRINT TABXY(59,4);Rev_vid$;L_blue$;" POINTS TO USE FOR ";Enh_off$;
4330      PRINT TABXY(59,5);Rev_vid$;L_blue$;"  MOVING AVERAGE  ";Enh_off$;
4332      PRINT TABXY(59,8);Rev_vid$;L_blue$;"  TRIGGER LEVEL   ";Enh_off$;
4334      RETURN
4336 !
4338 !*************************************************************************
4340 !
4342 Fill_in:     !
4344      GOSUB Print_volts
4346      GOSUB Print_offset
4348      GOSUB Print_freq
4350      GOSUB Print_save
4352      GOSUB Print_slope
4354      GOSUB Print_pairs
4356      GOSUB Print_sld_ave
```

```
4358      GOSUB Print_ave
4360      GOSUB Print_trig_lev
4362      RETURN
4364 !
4366 !*************************************************************************
4368 !
4370      !
4372      ! = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
4374      ! = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
4376      ! Data input subroutines    12/12/89
4378      ! = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
4380      ! = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
4382 !
4384 !*************************************************************************
4386 !
4388 Tcal_set:      !
4390                !this allows the operator to manually set up the scope.
4392      OFF KEY
4394      OFF KBD
4396      OFF KNOB
4398      Tcal_intrpt=3
4400      CLEAR SCREEN
4402      PRINT TABXY(1,1);" This is to establish the delay of the time "
4404      PRINT " signal only. Any other desired changes will have to be "
4406      PRINT " accomplished using the waveform or tcal menus. Use a "
4408      PRINT " variable delay line to set the waveform to the desired "
4410      PRINT " position. If the delay is set using the DELAY key of the "
4412      PRINT " oscilloscope, the time calibration data delay time will be"
4414      PRINT " different than the device under test (DUT) waveform delay. "
4416      PRINT " Time calibration data is not valid for a DUT waveform "
4418      PRINT " which has a different delay time. "
4420      !
4422      PRINT TABXY(1,21);"The current channel is ",Chnnl
4424      INPUT "is this the correct channel ? Y/N ",Ch$
4426      IF (Ch$="n") OR (Ch$="N") THEN GOSUB Get_chn
4428      !
4430      CLEAR SCREEN
4432      PRINT TABXY(1,24);"Please set up the waveform and     "
4434      PRINT "press continue when done."
4436      LOCAL 707
4438      CALL Pause_key_on
4440      OFF KEY
4442      GOSUB Tcal_scope
4444      RETURN
4446 !
4448 !*************************************************************************
4450 !
4452 Chnl_err:  !
4454      BEEP
4456      DISP "Input out of range or disallowed value."
4458      WAIT Paused
4460      DISP "Try again ..."
4462 Get_chn:     !
4464      ON ERROR GOTO Chnl_err
4466      Test$=""
4468      INPUT "Enter the acquisition channel number (1-4).",Test$
```

```
4470      IF LEN(Test$)<1 THEN RETURN
4472      Temp=VAL(Test$)
4474      OFF ERROR
4476      IF (Temp<1) OR (Temp>4) THEN GOTO Chnnl_error
4478      OUTPUT @Scope;"BLANK CHANNEL"&VAL$(Chnnl)
4480      Chnnl=Temp
4482      OUTPUT @Scope;"VIEW CHANNEL"&VAL$(Chnnl)
4484      ! The number of point can only be changed from the controller.
4486      RETURN
4488 !
4490 !*************************************************************************
4492 !
4494 Volterr: !
4496      GOSUB Beeps
4498      DISP "ERROR IN READING VALUE OR VALUE OUT OF RANGE, TRY AGAIN."
4500      WAIT Paused
4502 Input_volts:      !
4504      ON ERROR GOSUB Volterr
4506      Test$=""
4508      INPUT "Enter the volts per division in volts ...",Test$
4510      IF LEN(Test$)<1 THEN RETURN
4512      GOSUB Data_check
4514      IF (Rtemp<(Atten*.001)) OR (Rtemp>(Atten*.080)) THEN GOTO Volterr
4516      Tc_volt=Rtemp
4518      Tc_vrange=Tc_volt*8
4520      OUTPUT 707;":CHANNEL"&VAL$(Chnnl)&":RANGE "&VAL$(Tc_vrange)
4522      OFF ERROR
4524 Print_volts:      !
4526      PRINT TABXY(1,5);Orange$;
4528      CALL Auto_format(Tc_volt)
4530      IF NOT (Tcal_intrpt) THEN CALL Select_graphics(T_which,Tlast,Co_ords(*))
4532      RETURN
4534 !
4536 !*************************************************************************
4538 !
4540 Offset_err: !
4542      GOSUB Beeps
4544      DISP "ERROR IN READING VALUE OR VALUE OUT OF RANGE, TRY AGAIN."
4546      WAIT Paused
4548 Input_offset:      !
4550      ON ERROR GOTO Offset_err
4552      Test$=""
4554      INPUT "Enter the offset in volts ...",Test$
4556      IF LEN(Test$)<1 THEN RETURN
4558      GOSUB Data_check
4560      IF ABS(Rtemp)>5.00E-1 THEN GOTO Offset_err
4562      Tc_off=Rtemp
4564      OUTPUT 707;":CHANNEL"&VAL$(Chnnl)&":OFFSET "&VAL$(Tc_off)
4566      OFF ERROR
4568 Print_offset:!
4570      PRINT TABXY(1,8);Orange$;
4572      CALL Auto_format(Tc_off)
4574      IF NOT (Tcal_intrpt) THEN CALL Select_graphics(T_which,Tlast,Co_ords(*))
4576      RETURN
4578 !
4580 !*************************************************************************
```

```
4582 !
4584 Choose_average:      !
4586      OFF KEY
4588      OFF KNOB
4590      OFF KBD
4592      CLEAR SCREEN
4594      GCLEAR
4596      PRINT L_blue$
4598      DIM Dp$[80]
4600      DIM T$[52]
4602      Dp$="Select Average "
4604      T$="Available Averages (powers of 2 )    "
4606      Intr_prty=Intr_prty+3
4608      CALL Menu_scroll(Dp$,T$,Allowd$(*),12,1,What(*))
4610      Intr_prty=Intr_prty-3
4612      IF What(1)<>0 THEN ! Aborted
4614         T_aver=VAL(Allowd$(What(1)))
4616      END IF
4618      Tcal_intrpt=3
4620      !
4622      OUTPUT @Scope;":ACQUIRE:TYPE AVERAGE"
4624      OUTPUT @Scope;":ACQUIRE:COUNT "&VAL$(T_aver)
4626      RETURN
4628 Print_ave:      !
4630      PRINT TABXY(63,2);Orange$;
4632      CALL Auto_format(T_aver*1.0)
4634      RETURN
4636 !
4638 !*************************************************************************
4640 !
4642 Save_wave:      !
4644      IF Save$="  YES    " THEN
4646         Save$="   NO    "
4648      ELSE
4650         Save$="  YES    "
4652      END IF
4654 Print_save:      !
4656      PRINT TABXY(36,2);Orange$;
4658      PRINT Save$
4660      IF NOT (Tcal_intrpt) THEN CALL Select_graphics(T_which,Tlast,Co_ords(*))
4662      RETURN
4664 !
4666 !*************************************************************************
4668 !
4670 Freq_error: !
4672      BEEP
4674      DISP "ERROR FREQUENCY INPUT, TRY AGAIN."
4676      WAIT Paused
4678 Input_freq: !
4680      ON ERROR GOTO Freq_error
4682      Test$=""
4684      INPUT "Enter the calibration frequency...",Test$
4686      IF LEN(Test$)<1 THEN RETURN
4688      GOSUB Data_check
4690      OFF ERROR
4692      IF Rtemp<0 THEN GOTO Freq_error
```

```
4694      Freq=Rtemp
4696 Print_freq:          !
4698      PRINT TABXY(2,2);Orange$;
4700      CALL Auto_format(Freq)
4702      IF NOT (Tcal_intrpt) THEN CALL Select_graphics(T_which,Tlast,Co_ords(*))
4704      RETURN
4706 !
4708 !*************************************************************************
4710 !
4712 Pair_error: !
4714      GOSUB Beeps
4716      DISP "ERROR IN READING VALUE OR DISALLOWED VALUE, TRY AGAIN."
4718      WAIT Paused
4720 Point_pairs: !
4722      ON ERROR GOTO Pair_error
4724      Test$=""
4726      DISP "Enter the point pairs for the linear least squares fit."
4728      WAIT Paused
4730      INPUT " This must be a positive, integer number.",Test$
4732      IF LEN(Test$)<1 THEN RETURN
4734      Temp=VAL(Test$)
4736      OFF ERROR
4738      IF Temp<1 THEN GOTO Pair_error
4740      Ls_prs=Temp
4742 Print_pairs:      !
4744      PRINT TABXY(35,9);Orange$;
4746      CALL Auto_format(Ls_prs*1.0)
4748      IF NOT (Tcal_intrpt) THEN CALL Select_graphics(T_which,Tlast,Co_ords(*))
4750      RETURN
4752 !
4754 !*************************************************************************
4756 !
4758 Which_slope: !
4760      IF Slope THEN
4762         Slpe$=" - SLOPE "
4764         Slope=0
4766      ELSE
4768         Slpe$=" + SLOPE "
4770         Slope=1
4772      END IF
4774 Print_slope:      !
4776      PRINT TABXY(35,5);Orange$;
4778      PRINT Slpe$
4780      IF NOT (Tcal_intrpt) THEN CALL Select_graphics(T_which,Tlast,Co_ords(*))
4782      RETURN
4784 !
4786 !*************************************************************************
4788 !
4790 Moving_error: !
4792      GOSUB Beeps
4794      DISP "ERROR IN READING VALUE OR DISALLOWED VALUE, TRY AGAIN."
4796      WAIT Paused
4798      DISP "This must be an odd, positive integer number."
4800      WAIT Paused
4802 Moving_points: !
4804      ON ERROR GOTO Moving_error
```

```
4806      Test$ = ""
4808      INPUT "Enter an odd  number of points for the moving average.",Test$
4810      IF LEN(Test$)<1 THEN RETURN
4812      Temp = VAL(Test$)
4814      OFF ERROR
4816      IF Temp<1 THEN GOTO Moving_error
4818      IF NOT (Temp MOD 2) THEN GOTO Moving_error
4820      Zero_x = Temp
4822 Print_sld_ave:    !
4824      PRINT TABXY(63,6);Orange$;
4826      CALL Auto_format(Zero_x*1.0)
4828      IF NOT (Tcal_intrpt) THEN CALL Select_graphics(T_which,Tlast,Co_ords(*))
4830      RETURN
4832 !
4834 !*************************************************************************
4836 !
4838 Trig_lev_err:              !
4840      GOSUB Beeps
4842      DISP "ERROR IN READING DELAY, TRY AGAIN."
4844      WAIT Paused
4846 Trigger_lvl:    !
4848      ON ERROR GOTO Trig_lev_err
4850      Test$ = ""
4852      INPUT "Enter the desired trigger level in volts.",Test$
4854      IF LEN(Test$)<1 THEN RETURN
4856      GOSUB Data_check
4858      IF (Rtemp<-1.) OR (Rtemp>1.) THEN
4860         GOSUB Beeps
4862         DISP "VALUE OUT OF RANGE. Please try again."
4864         WAIT 1.0
4866         GOTO Trigger_lvl
4868      END IF
4870      T_trig = Rtemp
4872      OUTPUT @Scope;":TRIGGER:LEVEL "&VAL$(T_trig)
4874      OFF ERROR
4876 Print_trig_lev:              !
4878      PRINT TABXY(60,9);Orange$;
4880      CALL Auto_format(T_trig)
4882      IF NOT (Tcal_intrpt) THEN CALL Select_graphics(T_which,Last1,Co_ords(*))
4884      RETURN
4886 !
4888 !*************************************************************************
4890 !
4892 Tcal_data: !
4894              !
4896   !
4898   ! Time_data is stored as follows:
4900   ! Time_data(i,1) = time of zero crossing,
4902   ! Time_data(i,2) = point number of zero crossing,
4904   ! Time_data(kount + 1,1) = Actual time window,
4906   ! Time_data(kount + 2,1) = scope time window (10 * time_per_div),
4908   ! Time_data(kount + 3,1) = kount (number of zero crossings found),
4910   ! Time_data(kount + 4,1) = number of points acquired.
4912   !
4914      Kill_meas = 0
4916      Do_meas = 1
```

```
4918      OFF KEY
4920      OFF KBD
4922      OFF KNOB
4924      WHILE NOT (Kill_meas) AND (Do_meas)
4926         ON KEY 0 LABEL "ABORT ",Local_prty+5 GOTO Abort_tmeas
4928         IF Do_meas THEN GOSUB Get_twave
4930      END WHILE
4932      IF NOT (Kill_meas) THEN
4934         GOSUB Con_to_volts
4936         GOSUB Sliding_ave
4938         GOSUB Linear_fit
4940         GOSUB Crossing_point
4942         IF Save$="  YES    " THEN
4944            DISP "A description of the time calibration waveform required;"
4946            WAIT Paused
4948            INPUT "enter data description, 40 chrs or less. ",Data_id$
4950            Intr_prty=Local_prty+1
4952            CALL Data_to_disk_r(1,INT(Pnts),Voltage(*),Data_id$)
4954            Intr_prty=Local_prty
4956         END IF
4958         OFF KEY
4960         CLEAR SCREEN
4962         PRINT TABXY(1,1);"There were ";Time_data(Kount+3,1);
4964         PRINT "zero crossings and ";Time_data(Kount+4,1);"points acquired."
4966         PRINT "The time window as calculated against the time standard ";
4968         PRINT "is ";Time_data(Kount+1,1);"."
4970         PRINT "The time window that the oscilloscope reports is ";Time_data(Kount+2,1);"."
4972         INPUT "Is the data Okay? y/n",Ch$
4974         IF Ch$="n" OR Ch$="N" THEN
4976            DISP "Please re-set the measurement and try again."
4978            WAIT 1.5
4980            CLEAR @Scope
4982         ELSE
4984            DISP "Enter in the description of the time ";
4986            INPUT "calibration data, 40 characters or less. ",Data_id$
4988            Intr_prty=Local_prty+1
4990            CALL Data_to_disk_r(1,Kount+4,Time_data(*),Data_id$)
4992            Intr_prty=Local_prty
4994         END IF
4996         DEALLOCATE Voltage(*)
4998         DEALLOCATE Xcross(*)
5000         DEALLOCATE Time_data(*)
5002         DEALLOCATE Xpnts(*)
5004      END IF
5006      Tcal_intrpt=3
5008      OFF KEY
5010      RETURN
5012 !
5014 !*****************************************************************
5016 !
5018 Abort_tmeas: !
5020      CLEAR @Scope
5022      LOCAL @Scope
5024      Kill_meas=1
5026      CLEAR SCREEN
5028      DISP Red$;"MEASUREMENT ABORTED"
```

```
5030      OFF KEY
5032      OFF KNOB
5034      OFF KBD
5036      OFF ERROR
5038      OFF TIMEOUT 7
5040      Tcal_intrpt=3
5042      GOTO Tcal_scope
5044  !
5046  !*************************************************************************
5048  !
5050 Data_error:      !
5052      GOSUB Beeps
5054      DISP "ERROR IN READING SCOPE. WILL TRY AGAIN"
5056 Get_twave:      !
5058      Do_meas=0
5060      Try_again=1
5062 Get_again:      !
5064      IF NOT (Kill_meas) THEN
5066         ON ERROR GOTO Data_error
5068         OUTPUT @Scope;"*CLS"
5070         WAIT 1
5072         OUTPUT @Scope;":TER?"
5074         ENTER @Scope;Ter$
5076         IF (VAL(Ter$)<>1) THEN
5078            GOSUB Beeps
5080            DISP "No signal detected; please check the setup."
5082            WAIT 2
5084            CLEAR SCREEN
5086            OFF KEY
5088            OFF KNOB
5090            OFF KBD
5092            OFF ERROR
5094            CALL Scope_init(Err_flg)
5096            IF Err_flg THEN Interrupted=1
5098            Tcal_intrpt=3
5100            SUBEXIT
5102         END IF
5104         DISP "System busy...."
5106         OUTPUT @Scope;":TIMEBASE:DELAY "&VAL$(Dly)
5108         OUTPUT @Scope;":TIMEBASE:RANGE "&VAL$(Trange)
5110         OUTPUT @Scope;":CHANNEL"&VAL$(Chnnl)&":OFFSET  "&VAL$(Tc_off)
5112         OUTPUT @Scope;":CHANNEL"&VAL$(Chnnl)&":RANGE  "&VAL$(Tc_vrange)
5114         OUTPUT @Scope;":ACQUIRE:TYPE AVERAGE"
5116         OUTPUT @Scope;":ACQUIRE:COUNT "&VAL$(T_aver)
5118         OUTPUT @Scope;":ACQUIRE:POINTS "&VAL$(Pnts)
5120         OUTPUT @Scope;"*CLS"
5122         OUTPUT @Scope;"*SRE 32;*ESE 1"
5124         OUTPUT @Scope;":DIGITIZE CHAN"&VAL$(Chnnl)&";*OPC"
5126         !
5128         ! The following loop enables a keyboard abort key to be
5130         ! processed.
5132         ! The ABORT command is processed in the Tcal_data block.
5134         WHILE NOT BIT(Stat,5)
5136            Stat=SPOLL(@Scope)
5138         END WHILE
5140         OUTPUT @Scope;":SYSTEM:HEADER OFF;:EOI ON"
```

```
5142        OUTPUT @Scope;"WAVEFORM:SOURCE WMEMORY"&VAL$(Chnnl)&"; FORMAT WORD"
5144        OUTPUT @Scope;"WAVEFORM:DATA?"
5146        ENTER @Scope USING "#,A,D";Header$,Bytes
5148        IF Bytes=3 THEN
5150           ENTER @Scope USING "#,3D";Length
5152        END IF
5154        IF Bytes=4 THEN
5156           ENTER @Scope USING "#,4D";Length
5158        END IF
5160        Length=Length/2
5162        IF Pnts<>Length THEN
5164           DISP "The scope will not allow ",Pnts,"points."
5166           WAIT Paused
5168           DISP "The number of points is now ",Length,"."
5170           WAIT Paused
5172           Pnts=Length
5174        END IF
5176        REDIM Wav(Length)
5178        ENTER @Scope USING "#,W";Wav(*)
5180        ENTER @Scope USING "-K,B";End$
5182        OUTPUT @Scope;":WAVEFORM:YINCREMENT?"
5184        ENTER @Scope;Yinc
5186        OUTPUT @Scope;":WAVEFORM:YORIGIN?"
5188        ENTER @Scope;Yorg
5190        OUTPUT @Scope;":WAVEFORM:YREFERENCE?"
5192        ENTER @Scope;Yref
5194        OUTPUT @Scope;":WAVEFORM:XINCREMENT?"
5196        ENTER @Scope;Xinc
5198        OUTPUT @Scope;":WAVEFORM:XORIGIN?"
5200        ENTER @Scope;Xorg
5202        OUTPUT @Scope;":WAVEFORM:XREFERENCE?"
5204        ENTER @Scope;Tref
5206        OFF ERROR
5208 !      Bug1=1
5210        IF Bug1 THEN
5212           FOR I=1 TO Pnts
5214              PRINTER IS PRT
5216              PRINT Wav(I)," ",I
5218           NEXT I
5220           PRINTER IS CRT
5222           Bug1=0
5224        END IF
5226     END IF
5228     RETURN
5230 !
5232 !*************************************************************************
5234 !
5236 Con_to_volts: !
5238     ALLOCATE Voltage(Pnts,2)
5240     FOR I=1 TO Pnts
5242        Voltage(I,2)=((Wav(I)-Yref)*Yinc)+Yorg
5244        Voltage(I,1)=((I-Tref)*Xinc)+Xorg
5246     NEXT I
5248     RETURN
5250 !
5252 !*************************************************************************
```

```
5254 !
5256 Sliding_ave: !
5258     ALLOCATE REAL Xcross(100)
5260     ALLOCATE Xpnts(Pnts)
5262     Kount=0
5264     Int_temp=Zero_x DIV 2
5266     GOSUB Mid
5268     Zero_level=(Midval*1.0)*(Zero_x*1.0)
5270     Slide_ave=0.-Zero_level
5272     FOR I=1 TO Zero_x
5274        Xpnts(I)=I
5276        Slide_ave=Slide_ave+Wav(I)
5278     NEXT I
5280     FOR I=Zero_x+1 TO Pnts
5282        Xpnts(I)=I
5284        Prev_ave=Slide_ave
5286        Slide_ave=Slide_ave+Wav(I)-Wav(I-Zero_x)
5288        IF ((Prev_ave*Slide_ave<0) OR (Slide_ave=0)) THEN
5290     ! a change in sign indicates a zero crossing.
5292     !
5294     !Negative slope crossing
5296     !
5298           IF (NOT Slope) AND ((Slide_ave-Prev_ave)<=0.) THEN
5300              Kount=Kount+1
5302              Xcross(Kount)=I-Int_temp
5304           END IF
5306     !
5308     !Positive slope crossing
5310     !
5312           IF (Slope) AND ((Slide_ave-Prev_ave)>=0.) THEN
5314              Kount=Kount+1
5316              Xcross(Kount)=I-Int_temp
5318           END IF
5320        END IF
5322     NEXT I
5324     RETURN
5326 !
5328 !*************************************************************************
5330 !
5332 Mid:  !
5334     Maxval=Wav(1)*1.0
5336     Minval=Wav(1)*1.0
5338     FOR I=1 TO Pnts
5340        IF (Wav(I)*1.0)>Maxval THEN Maxval=Wav(I)*1.0
5342        IF (Wav(I)*1.0)<Minval THEN Minval=Wav(I)*1.0
5344     NEXT I
5346     Midval=INT((Maxval+Minval)/2)
5348     RETURN
5350 !
5352 !*************************************************************************
5354 !
5356 Crossing_point: !
5358           !
5360     ALLOCATE Time_data(Kount+4,2)
5362   !
5364   ! Time_data is stored as follows:
```

```
5366   ! Time_data(i,1)=time of zero crossing,
5368   ! Time_data(i,2) = point number of zero crossing,
5370   ! Time_data(kount+1)= Actual time window,
5372   ! Time_data(kount+2) = scope time window (10 * time_per_div),
5374   ! Time_data(kount+3) = kount (number of zero crossings found),
5376   ! Time_data(kount+4) = number of points acquired.
5378   !
5380     INTEGER K
5382     Period=1/Freq
5384     Scale_fact=Period/(Xcross(2)-Xcross(1)) ! Scale_fact is the number
5386                                     ! for converting point numbers to
5388                                     !a time value.
5390     Time_data(1,1)=Xcross(1)*Scale_fact
5392     Time_data(1,2)=Xcross(1)
5394     FOR K=2 TO Kount
5396        Scale_fact=Period/(Xcross(K)-Xcross(K-1))
5398        Time_data(K,1)=Xcross(K)*Scale_fact
5400        Time_data(K,2)=Xcross(K)
5402        IF Bug1 THEN
5404           PRINTER IS PRT
5406           PRINT "time_data(",K,",1) = ",Time_data(K,1)
5408           PRINT "time_data(",K,",2) = ",Time_data(K,2)
5410           PRINTER IS CRT
5412        END IF
5414     NEXT K
5416     Time_data(Kount+1,1)=Time_data(Kount,1)+Scale_fact*(Pnts-Xcross(Kount))
5418     Time_data(Kount+2,1)=Time_per_div*10
5420     Time_data(Kount+3,1)=Kount
5422     Time_data(Kount+4,1)=Pnts
5424     RETURN
5426   !
5428   !*************************************************************************
5430   !
5432 Linear_fit: !
5434     ALLOCATE B(2,1)
5436          ! B(1,1) is the intercept, B(2,1) is the slope.
5438          ! of the line which best fits the data around a
5440          ! particular data point.
5442     INTEGER Xval
5444     IF (Xcross(1)<Ls_prs) THEN
5446        GOSUB Beeps
5448        DISP "The initial zero crossing is too close to the left"
5450        WAIT Paused
5452        DISP "side of the scope display and has been omitted."
5454        WAIT Paused
5456        FOR I=1 TO Kount-1
5458           Xcross(I)=Xcross(I+1)
5460        NEXT I
5462        Kount=Kount-1
5464     END IF
5466     IF ((Pnts-Xcross(Kount))<Ls_prs) THEN
5468        DISP "The last zero crossing is too close to the right"
5470        WAIT Paused
5472        DISP "side of the scope display and has been omitted."
5474        Kount=Kount-1
5476     END IF
```

```
5478      FOR I=1 TO Kount
5480         Xval=Xcross(I)-Ls_prs
5482         CALL Straight_line(Xval,INT(Ls_prs),B(*))
5484         IF B(2,1)<>0 THEN
5486            Xcross(I)=((1.0*Midval)-B(1,1))/B(2,1)
5488         END IF
5490      NEXT I
5492      DEALLOCATE B(*)
5494      RETURN
5496  !
5498  !*************************************************************************
5500  !
5502 Data_check: !
5504      ! The following is a test of the lower case e in a number of
5506      ! scientific notation. If a lower case e occurs, it is converted
5508      ! to upper case. That is all.
5510      IF POS(Test$,"e") THEN
5512         Temp=POS(Test$,"e")
5514         Test$[Temp]="E"&Test$[Temp+1,LEN(Test$)]
5516      END IF
5518      ! end of lower case conversion
5520      Rtemp=VAL(Test$)
5522      RETURN
5524  !
5526  !*************************************************************************
5528  !
5530 Beeps:      !
5532      BEEP 400,.25
5534      BEEP 600,.50
5536      BEEP 400,.25
5538      RETURN
5540      !
5542   SUBEND
5544   !
5546   ! ***********************************************************************
5548   !
5550   SUB Straight_line(INTEGER Xval,Ls_points,REAL B(*))
5552 Straight_line:  !
5554      OPTION BASE 1
5556      COM /Line_fit/ INTEGER Wav(1024)
5558      DEG
5560      ALLOCATE REAL Mtrx(Ls_points,2),Transps(2,Ls_points),Temp_mat(2,2)
5562      ALLOCATE REAL Temp_str(2,Ls_points),Vectr(Ls_points)
5564      INTEGER Half_pnts
5566      !
5568      Half_pnts=Ls_points DIV 2
5570      FOR I=1 TO Ls_points
5572         Mtrx(I,1)=1
5574         Mtrx(I,2)=Xval-Half_pnts+I-1
5576         Vectr(I)=Wav(Xval-Half_pnts+I-1)
5578      NEXT I
5580      MAT Transps= TRN(Mtrx)
5582      MAT Temp_mat= Transps*Mtrx
5584      MAT Temp_mat= INV(Temp_mat)
5586      MAT Temp_str= Temp_mat*Transps
5588      MAT B= Temp_str*Vectr
```

```
5590      SUBEXIT
5592    SUBEND
5594    !
5596    !*************************************************************************
5598    !
5600    SUB Pause_key_on
5602  Pause_key_on: ! Make sure that CONTINUE key exists.
5604            ! Original: 02 Dec 1987
5606            ! Revision: 02 Dec 1987
5608      COM /Sys/ Sys_id$[10]
5610      IF Sys_id$[1,4]="S300" THEN ! reset to S300 system keys
5612        CONTROL KBD,15;0
5614        CONTROL CRT,12;2
5616        LOAD KEY
5618      END IF
5620      PAUSE
5622      IF Sys_id$[1,4]="S300" THEN  ! set to S200 compatible keys
5624        OUTPUT KBD USING "K,#";"SCRATCH KEYX"
5626        CONTROL KBD,15;1
5628        CONTROL CRT,12;0
5630      END IF
5632      SUBEXIT
5634    SUBEND
5636    !
5638    !*************************************************************************
5640    SUB Auto_format(Value)
5642  Auto_format: ! Original: 13 Nov 1984
5644            ! Revision: 06 Aug 1987
5646      ! Select the proper number of digits to display.
5648      ! This routine is used by several program sections to
5650      ! print numbers to the display.
5652      !
5654      SELECT ABS(Value)
5656      CASE 1.0 TO 99999.99
5658        IF Value=PROUND(Value,-2) THEN
5660          IF INT(Value)=Value THEN
5662            PRINT USING "#,M5D,6X";Value
5664          ELSE
5666            PRINT USING "#,M5D.DD,3X";Value
5668          END IF
5670        ELSE
5672          PRINT USING "#,MD.4DESZZ,X";Value
5674        END IF
5676        !
5678      CASE >99999.99
5680        PRINT USING "#,MD.3DESZZZ,X";Value
5682        !
5684        !+++++++All values less that 1.0 ++++++++++++
5686        !
5688      CASE .0001 TO 1
5690        IF PROUND(Value,-4)=Value THEN
5692          PRINT USING "#,4X,MZ.4D,X";Value
5694        ELSE
5696          PRINT USING "#,MD.4DESZZ,X";Value
5698        END IF
5700        !
```

```
5702      CASE 1.0E-99 TO .0001
5704         PRINT USING "#,MD.4DESZZ,X";Value
5706         !
5708      CASE 1.0E-300 TO 1.0E-99
5710         PRINT USING "#,MD.3DESZZZ,X";Value
5712      CASE ELSE
5714         PRINT USING "#,4X,MZ.D,4X";Value
5716      END SELECT
5718      SUBEXIT
5720   SUBEND
5722   !
5724   ! ***************************************************************
5726   !
5728   SUB Scope_init(INTEGER Err_flg)
5730 Scope_init:  !
5732            !
5734      COM /Scope/ REAL Time_per_div,Volts,Trange,Vrange,Dly
5736      COM /Scope/ REAL Probe_fac,Offs,Trig,Atten
5738      COM /Scope/ INTEGER Aver,Pnts,Chnnl
5740      COM /Scope/ Type$[30],Refer$[14],@Scope,Mode$[30]
5742      COM /Tcal_vals/ INTEGER Zero_x,Ls_prs,Slope,T_aver,REAL Freq
5744      COM /Tcal_vals/ Slpe$[10],Save$[10],REAL Tc_off,Tc_volt,T_trig
5746      COM /Vcal_vals/ INTEGER V_aver,Interval,REAL V_step,V_min
5748      !
5750      !
5752      Err_flg=0 !no errors yet.
5754      Try_again=1
5756      GOSUB Do_again
5758      SUBEXIT
5760 Init_err:     !
5762      BEEP
5764      DISP "Something went wrong during initialization. Will try again."
5766      WAIT 1.5
5768 Do_again:     !
5770      ON ERROR GOTO Init_err
5772      ON TIMEOUT 7,2 GOTO Time_out_error
5774      CLEAR 707 !clears the HPIB to the scope
5776      ASSIGN @Scope TO 707
5778      OUTPUT @Scope;"*RST" !Puts the scope in a known state.
5780      !The rest puts the scope in a more useful state.
5782      OUTPUT @Scope;"BLANK CHANNEL1"
5784      OUTPUT @Scope;"BLANK CHANNEL2"
5786      OUTPUT @Scope;"BLANK CHANNEL3"
5788      OUTPUT @Scope;"BLANK CHANNEL4"
5790      OUTPUT @Scope;":DISPLAY:GRATICULE GRID"
5792      OUTPUT @Scope;":DISPLAY:FORMAT 1"
5794      IF Refer$="   CENTER   " THEN
5796         OUTPUT @Scope;"TIMEBASE:REFERENCE CENTER"
5798      ELSE
5800         OUTPUT @Scope;"TIMEBASE:REFERENCE LEFT"
5802      END IF
5804      OUTPUT @Scope;":TIMEBASE:RANGE "&VAL$(Trange)
5806      OUTPUT @Scope;":TIMEBASE:DELAY "&VAL$(Dly)
5808      OUTPUT @Scope;":TIMEBASE:MODE "&Mode$
5810      OUTPUT @Scope;"VIEW CHANNEL"&VAL$(Chnnl)
5812      OUTPUT @Scope;":CHANNEL"&VAL$(Chnnl)&":OFFSET "&VAL$(Offs)
```

```
5814      OUTPUT @Scope;":CHANNEL"&VAL$(Chnnl)&":RANGE  "&VAL$(Vrange)
5816      OUTPUT @Scope;":ACQUIRE:TYPE "&Type$
5818      OUTPUT @Scope;":ACQUIRE:COUNT  "&VAL$(Aver)
5820      OUTPUT @Scope;":ACQUIRE:POINTS  "&VAL$(Pnts)
5822      OUTPUT @Scope;":ACQUIRE:BANDWIDTH  HIGH"
5824      OFF TIMEOUT 7
5826      OFF ERROR
5828      RETURN
5830 Time_out_error:      !
5832      IF Try_again>5 THEN
5834         OFF TIMEOUT 7
5836         OFF ERROR
5838         BEEP
5840         DISP "Scope not responding...command  aborted"
5842         WAIT 1
5844         Err_flg=1
5846         CLEAR SCREEN
5848         SUBEXIT
5850      END IF
5852      BEEP
5854      DISP "Scope not responding...will try again."
5856      WAIT 1.
5858      CLEAR SCREEN
5860      Try_again=Try_again+1
5862      GOTO Do_again
5864   SUBEND
5866      !
5868 !
5870 !*************************************************************************
5872 !
5874   SUB Select_graphics(INTEGER  Which,Last1,Co_ords(*))
5876 Select_graphics:      !
5878      IF Last1>0 THEN
5880         Lwrlftx=Co_ords(Last1,1)
5882         Lwrlfty=Co_ords(Last1,2)
5884         Upperrtx=Co_ords(Last1,3)
5886         Upperrty=Co_ords(Last1,4)
5888         PEN -1
5890         FOR I=1 TO 4
5892            MOVE Lwrlftx,Lwrlfty
5894            RECTANGLE Upperrtx,Upperrty
5896            Lwrlftx=Lwrlftx-2
5898            Lwrlfty=Lwrlfty-2
5900            Upperrtx=Upperrtx+4
5902            Upperrty=Upperrty+4
5904         NEXT I
5906      END IF
5908      Lwrlftx=Co_ords(Which,1)
5910      Lwrlfty=Co_ords(Which,2)
5912      Upperrtx=Co_ords(Which,3)
5914      Upperrty=Co_ords(Which,4)
5916      PEN 1
5918      FOR I=1 TO 4
5920         MOVE Lwrlftx,Lwrlfty
5922         RECTANGLE Upperrtx,Upperrty
5924         Lwrlftx=Lwrlftx-2
```

```
5926          Lwrlfty=Lwrlfty-2
5928          Upperrtx=Upperrtx+4
5930          Upperrty=Upperrty+4
5932       NEXT I
5934       Last1=Which
5936       SUBEXIT
5938    SUBEND
5940    !
5942    !*************************************************************************
5944    !
5946    SUB Operator_info
5948    !
5950 Operator_info: !
5952              !
5954       COM /Hue/ Rev_vid$[1],Enh_off$[1],Underline$[1]
5956       COM /Hue/ Red$[1],Orange$[1],L_blue$[1]
5958       !
5960       OFF KEY
5962       Interrupted=1
5964       CLEAR SCREEN
5966       PRINT L_blue$
5968       PRINT "This program performs automatic acquisition of waveforms"
5970       PRINT "for time domain analysis. "
5972       PRINT
5974       PRINT "There are potentially three data sets required per calibration,"
5976       PRINT "a waveform from the device under test (DUT), a time "
5978       PRINT "calibration waveform TCAL, and a set of voltage calibration "
5980       PRINT "waves VCAL (these are D.C. values). The number of "
5982       PRINT "calibration acquisitions performed is entirely "
5984       PRINT "up to the operator."
5986       PRINT
5988       PRINT "When this program is first loaded and run, the oscilloscope"
5990       PRINT "will be reset and initialized to 'default' values. "
5992       PRINT "There after, any changes made are saved as long as the "
5994       PRINT "program is in memory."
5996       PRINT
5998       PRINT "Press the continue key when ready to proceed."
6000       CALL Pause_key_on
6002       CLEAR SCREEN
6004       BEEP
6006       PRINT Red$
6008       PRINT "The scope MUST be set-up in the WAVEFORM MENU before"
6010       PRINT "using the time or voltage calibration acquisition routines."
6012       PRINT
6014       PRINT "The vertical scale is fixed irrespective of the screen "
6016       PRINT "display. This means a full waveform will be acquired even"
6018       PRINT "if the waveform appears clipped on the screen."
6020       PRINT "You can't always get what you want."
6022       PRINT L_blue$
6024       PRINT "Press the continue key when ready to proceed."
6026       CALL Pause_key_on
6028       CLEAR SCREEN
6030       PRINT "All values not selectable from a given menu are in "
6032       PRINT "memory. This is done to reduce the chance of acquiring"
6034       PRINT "data which has different time scales, voltage settings,"
6036       PRINT "offset level, e.t.c. If a value is changeable from a "
```

```
6038      PRINT "menu then it does not need to be consistent with the "
6040      PRINT "other waveforms. For example, the voltage offset level "
6042      PRINT "for the time calibration wave does not need to be the same"
6044      PRINT "as the offset level for the DUT wave; the timebase parameters"
6046      PRINT "are the important settings in this case. On the other hand, "
6048      PRINT "the offset must be equal for the DUT and the voltage "
6050      PRINT "calibration data sets. "
6052      PRINT
6054      PRINT "And so on."
6056      PRINT
6058      PRINT "All instrument connections are ,of course, done manually."
6060      PRINT
6062      PRINT "Press continue when you are ready proceed."
6064      PRINT Enh_off$
6066      CALL Pause_key_on
6068      CLEAR SCREEN
6070      SUBEXIT
6072 !
6074 !*************************************************************************
6076 !
6078   SUBEND
6080   SUB Data_to_disk_r(INTEGER Curve,Datacount,REAL Basket_file(*),Data_id$)
6082 Data_to_disk_r:  ! Original: 13 Nov 1984
6084                  ! Revision: 02 Dec 1987
6086      !This routine will SAVE data files on the disk in RAW data format.
6088      OPTION BASE 1
6090      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
6092      COM /Interrupts/ INTEGER Intr_prty
6094      INTEGER Local_prty,Diskspace
6096      DIM Ac$[5],Status$[1],Tempfile$[14]
6098      REAL Dtime
6100      OFF KEY
6102      Local_prty=Intr_prty
6104      Dtime=0.
6106      !
6108      !Select the disk drive for data storage
6110      !
6112 Selectdrive: !
6114      GRAPHICS OFF
6116      OUTPUT 2 USING "#,K";"K"
6118      CALL Select_disk
6120      IF Diskdrive$="NO DISK" THEN GOTO Mistakeline
6122 Choosefilename: !
6124      Ac$="ABORT"
6126      Tempfile$=Filename$
6128      CALL Enterfilename(Ac$)
6130      IF LEN(Filename$)=0 THEN
6132         Filename$=Tempfile$
6134         GOTO Mistakeline
6136      END IF
6138 Send_to_disk: ! Create file and save information.
6140      ON ERROR GOTO Cant_savedata
6142      Diskspace=INT((Datacount*16.0)/256)+2
6144      CREATE BDAT Ms_path$&Filename$&Diskdrive$,Diskspace,256
6146      Dtime=TIMEDATE
6148      DISP " SAVING data for CURVE # ";Curve;". "
```

```
6150      Status$ = "N"
6152      ASSIGN @Datapath TO Ms_path$&Filename$&Diskdrive$
6154      OUTPUT @Datapath;Status$
6156      OUTPUT @Datapath;Data_id$    !40 chrs description if single curve.
6158      OUTPUT @Datapath;Datacount   !number of xy points
6160      OUTPUT @Datapath;Datacount   !size of array (same as above)
6162      OUTPUT @Datapath;Basket_file(*)
6164      ASSIGN @Datapath TO *
6166      OFF ERROR
6168      !
6170 Mistakeline:OFF KEY
6172      LOOP
6174      EXIT IF TIMEDATE-Dtime>1.8
6176      END LOOP
6178      DISP CHR$(12)
6180      OUTPUT 2 USING "#,K";"K"
6182      SUBEXIT
6184      !
6186      ! ////////////////////////////////////////////////////
6188      !
6190 Cant_savedata:  !
6192      BEEP 500,.6
6194      SELECT ERRN
6196      CASE 72,73,76,78,81,82,90,93
6198         DISP Diskdrive$;" has failed or is not available ";
6200         DISP " ....CONTINUE to try again."
6202         CALL Pause_key_on
6204         Filename$ = Tempfile$
6206      CASE 84,85
6208         DISP " This disk is not initialized ";
6210         DISP " ....CONTINUE to try again."
6212         CALL Pause_key_on
6214         Filename$ = Tempfile$
6216      CASE 55,64
6218         DISP " This disk is full, insert new floppy and/or";
6220         DISP " select new drive ...CONTINUE "
6222         CALL Pause_key_on
6224         Filename$ = Tempfile$
6226      CASE ELSE
6228         CALL Errortrap
6230         GOTO Send_to_disk
6232      END SELECT
6234      GOTO Selectdrive
6236      !
6238   SUBEND
6240   !
6242   ! ***************************************************************************
6244   !
6246   SUB Select_disk
6248 Select_disk: ! Original: 13 Nov 1984
6250            ! Revision: 02 Dec 1987
6252      OPTION BASE 1
6254      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
6256      COM /Interrupts/ INTEGER Intr_prty
6258      COM /Sys_msi/ Msi_id$
6260      COM /Sys/ Sys_id$
```

```
6262    INTEGER Local_prty,Dd,Pt,Choose(1)
6264    DIM Disc$(30)[60],Title$[40],Displ$[60]
6266    Local_prty=Intr_prty
6268    OFF KEY
6270    !
6272    ! Define the disk drives available for this system, reserve the
6274    ! first characters for the drive address and the characters after
6276    ! the - for a description of the drive.
6278    !
6280    ! Example:
6282    ! Disc$(1)=":,700,0,0      HP 9133H HARD disk, volume 0."
6284    !
6286    !
6288    Displ$=" SELECT DISK DRIVE  ... Abort will cancel. "
6290    Title$=" Available disk drives for this system. "
6292    Pt=1   ! allow only one select
6294    !
6296    IF Diskdrive$[1,1]<>":" THEN Diskdrive$=""
6298    IF Msi_id$[1,1]<>":" THEN Msi_id$=SYSTEM$("MSI")
6300    IF Msi_id$[1,1]<>":" THEN ! Must be HFS subdirectory
6302       Ms_path$=Msi_id$[1,POS(Msi_id$,":")-1]  ! strip off subdirs
6304       IF Ms_path$[LEN(Ms_path$);1]<>"/" THEN Ms_path$=Ms_path$&"/"
6306       Msi_id$=Msi_id$[POS(Msi_id$,":"),LEN(Msi_id$)]
6308    END IF
6310    Diskdrive$=TRIM$(Diskdrive$)
6312    Msi_id$=TRIM$(Msi_id$)
6314    IF LEN(Diskdrive$)>0 AND LEN(Msi_id$)>0 THEN
6316       Disc$(1)=Diskdrive$&RPT$(" ",17-LEN(Diskdrive$))
6318       Disc$(1)=Disc$(1)&"- Last selected disk drive."
6320       Dd=1
6322       IF Diskdrive$<>Msi_id$ THEN
6324          Disc$(2)=Msi_id$&RPT$(" ",17-LEN(Msi_id$))
6326          Disc$(2)=Disc$(2)&"- Start-up mass storage unit specifier."
6328          Dd=Dd+1
6330       ELSE
6332          Disc$(1)=Disc$(1)&"  Start-up MSUS."
6334       END IF
6336    ELSE
6338       IF LEN(Msi_id$)>0 THEN
6340          Disc$(1)=Msi_id$&RPT$(" ",17-LEN(Msi_id$))
6342          Disc$(1)=Disc$(1)&"- Start-up mass storage unit specifier."
6344          Dd=1
6346       ELSE
6348          Dd=0
6350       END IF
6352    END IF
6354 Disk: !
6356    ! ..... customize system drives here .....
6358    ! Follow format with - after unit specifier, description is
6360    ! optional but recommended.
6362    ! ..........................................................
6364    !
6366    Disc$(Dd+1)=":,702,0        - HP 9122 dual microfloppy left drive"
6368    Disc$(Dd+2)=":,702,1        - HP 9122 dual microfloppy right drive"
6370    Disc$(Dd+3)=":,703,0        - HP 9125 single 5.25 floppy drive"
6372    Disc$(Dd+4)=":,1400         - HP 9133H hard disk volume 1"
```

```
6374      !
6376      Dd=Dd+4    ! add the number of drive specifiers above
6378      !
6380      IF Sys_id$[1,4]<>"S300" THEN
6382         Disc$(Dd+1)=":,4,1          - LEFT internal series 200"
6384         Disc$(Dd+2)=":,4,0          - RIGHT internal series 200"
6386         Dd=Dd+2
6388      END IF
6390      ! ..........................................................
6392      !
6394      Intr_prty=Local_prty+1
6396      CALL Menu_scroll(Displ$,Title$,Disc$(*),Dd,Pt,Choose(*))
6398      Intr_prty=Local_prty-1
6400      IF Pt=0 THEN
6402         Diskdrive$="NO DISK"
6404      ELSE
6406         Dd=POS(Disc$(Choose(Pt)),"-")-1  ! find -
6408         IF Dd>5 THEN  !  valid msus
6410            Diskdrive$=TRIM$(Disc$(Choose(Pt))[1,Dd])
6412         ELSE
6414            DISP " ERROR in reading MSUS from string, - chr not found. "
6416            BEEP
6418            CALL Pause_key_on
6420            Diskdrive$="NO DISK"
6422         END IF
6424      END IF
6426 Diskselected:OFF KEY
6428      SUBEXIT
6430   SUBEND
6432   !
6434   ! ************************************************************
6436   !
6438   SUB Enterfilename(Ac$)
6440 Enterfilename:  ! Original: 13 Nov 1984
6442              ! Revision: 10 Dec 1990  includes HFS directories
6444      OPTION BASE 1
6446      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
6448      COM /Interrupts/ INTEGER Intr_prty
6450      INTEGER I,Ascii_num,Maskflag,Namelength
6452      DIM Test$[256],Hfs_temp$[161]
6454      Namelength=10
6456      IF LEN(Ms_path$)>0 THEN OUTPUT KBD USING "K,#";"#"&Ms_path$&"H"
6458      DISP " ENTER HFS directory PATH (no file)";
6460      IF Ac$<>"PATH" THEN
6462         DISP ", ENTER / for HFS ROOT or null for LIF...";
6464      END IF
6466      LINPUT Hfs_temp$
6468      Hfs_temp$=TRIM$(Hfs_temp$)
6470      IF LEN(Hfs_temp$)>0 THEN
6472         IF LEN(Hfs_temp$)>1 AND Hfs_temp$[LEN(Hfs_temp$);1]<>"/" THEN
6474            Hfs_temp$=Hfs_temp$&"/"
6476         END IF
6478         IF LEN(Hfs_temp$)=1 THEN Hfs_temp$=""
6480         Namelength=14
6482      END IF
6484      IF Ac$="PATH" THEN
```

```
6486        Ms_path$=Hfs_temp$
6488        SUBEXIT
6490      END IF
6492      IF LEN(Filename$)>0 THEN OUTPUT KBD USING "K,#";"#"&Filename$&"H"
6494 Efn: !
6496      DISP " ENTER the FILE NAME ... ";
6498      SELECT Ac$
6500      CASE "CAT"
6502        DISP "(ENTER CAT mask* or ENTER null to CAT)";
6504      CASE "ABORT"
6506        DISP "(ENTER null to ABORT) ";
6508      CASE "VALID"
6510        DISP "(must be a VALID name!) ";
6512      END SELECT
6514      LINPUT Test$
6516      Test$=TRIM$(Test$)
6518      IF LEN(Test$)=0 AND Ac$="VALID" THEN GOTO Enterfilename
6520      IF LEN(Test$)=0 THEN Abortline
6522      IF LEN(Test$)>Namelength THEN
6524        BEEP
6526        DISP "ERROR in NAME ENTRY - max ";Namelength;" chars, you have ";
6528        DISP LEN(Test$);" "
6530        WAIT 1.8
6532        OUTPUT 2 USING "K,#";"#"&Test$&"H"
6534        GOTO Efn
6536      END IF
6538      IF POS(Test$,"*")>1 THEN
6540        Test$=Test$[1,POS(Test$,"*")-1]
6542        Maskflag=1
6544      ELSE
6546        Maskflag=0
6548      END IF
6550      FOR I=1 TO LEN(Test$)
6552        Ascii_num=NUM(Test$[I])
6554        SELECT Ascii_num
6556        CASE 65 TO 90,95,97 TO 122,48 TO 57
6558          !Allowed characters
6560        CASE ELSE
6562          BEEP
6564          DISP "ERROR in NAME ENTRY--ILLEGAL CHARACTERS, TRY AGAIN."
6566          WAIT 1.8
6568          OUTPUT 2 USING "K,#";"#"&Test$&"H"
6570          GOTO Efn
6572        END SELECT
6574      NEXT I
6576      IF Maskflag THEN
6578        Filename$=Test$&"*"
6580      ELSE
6582        Filename$=Test$
6584      END IF
6586      Ms_path$=Hfs_temp$
6588      SUBEXIT
6590 Abortline:Filename$=""
6592      IF Ac$="CAT" THEN Ms_path$=Hfs_temp$
6594      SUBEXIT
6596    SUBEND
```

```
6598  !
6600  ! ************************************************************
6602  !
6604  SUB File_menu(Mask$,Ftype$,Fls$(*),INTEGER Fls_cnt,Dir_on,Prt_on)
6606 File_menu:  !
6608     ! Original: 29 Jun 1987, G. Koepke
6610     ! Revision: 02 Dec 1987, 07:00
6612     OPTION BASE 1
6614     DEG
6616     COM /Sys/ Sys_id$[10]
6618     COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
6620     COM /Interrupts/ INTEGER Intr_prty
6622     COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
6624     DIM Directory$(600)[80],Bd$(600)[71]
6626     DIM D$[80],T$[51],Ids$[40],Stat$[1],Test$[256]
6628     INTEGER Bd_cnt,File_cnt,I,C_cnt,C0(1),Format_error,End_search
6630     IF Fls_cnt>0 THEN ALLOCATE INTEGER Choose(Fls_cnt)
6632     !
6634     ! Catalog the disk specified
6636     !
6638     End_search=0
6640     REPEAT   ! Generate path to file and extract file name.
6642        ON ERROR GOTO Cat_errors
6644        DISP " Reading the Directory ... "
6646        IF LEN(Ms_path$)>0 THEN
6648           MASS STORAGE IS Ms_path$[1,LEN(Ms_path$)-1]&Diskdrive$
6650        ELSE
6652           MASS STORAGE IS Diskdrive$
6654        END IF
6656        CAT TO Directory$(*);NO HEADER,COUNT File_cnt
6658        OFF ERROR
6660        !
6662        ! set up array of legal file names.
6664        !
6666        Bd_cnt=0
6668        MAT Bd$= ("")
6670        FOR I=1 TO File_cnt
6672           SELECT Directory$(I)[32,36]
6674           CASE Ftype$          ! Ftype$ = "BDAT " or
6676                                ! Ftype$ = "PROG "
6678              IF LEN(Mask$)>0 THEN ! Test for mask$
6680                 IF Directory$(I)[1,LEN(Mask$)]=Mask$ THEN
6682                    Bd_cnt=Bd_cnt+1
6684                    Bd$(Bd_cnt)=Directory$(I)[1;14]&" - "&Ftype$
6686                 END IF
6688              ELSE
6690                 Bd_cnt=Bd_cnt+1
6692                 Bd$(Bd_cnt)=Directory$(I)[1;14]&" - "&Ftype$
6694              END IF
6696           CASE "DIR "          ! plus all "DIR " listings
6698              Bd_cnt=Bd_cnt+1
6700              Bd$(Bd_cnt)=Directory$(I)[1;14]&" - DIR "
6702           CASE ELSE
6704           END SELECT
6706        NEXT I
6708        IF LEN(Ms_path$)>0 AND Bd_cnt>0 AND Fls_cnt>0 THEN
```

```
6710        Bd_cnt=Bd_cnt+1
6712        Bd$(Bd_cnt)="----  MOVE back up ONE Directory level."
6714        Bd_cnt=Bd_cnt+1
6716        Bd$(Bd_cnt)="----  RETURN to ROOT Directory."
6718      END IF
6720      !
6722      ! set up file menu
6724      !
6726      D$=" Select "&VAL$(Fls_cnt)&" file name(s) for data entry."
6728      T$="List of "&Ftype$&"files and DIRs on "&Diskdrive$
6730      IF LEN(Mask$)>0 THEN
6732        T$=T$&" mask="&Mask$
6734      END IF
6736      IF Bd_cnt>0 THEN
6738        IF Dir_on>0 THEN GOSUB Read_data_id
6740        IF Prt_on THEN
6742          GOSUB List_directory
6744          End_search=1
6746        ELSE
6748          C_cnt=Fls_cnt
6750          DISP CHR$(12)
6752          IF Fls_cnt>0 THEN
6754            CALL Menu_scroll(D$,T$,Bd$(*),Bd_cnt,C_cnt,Choose(*))
6756          ELSE
6758            CALL Menu_scroll(D$,T$,Bd$(*),Bd_cnt,C_cnt,C0(*))
6760          END IF
6762          !
6764          ! transfer file names to Fls$(*).
6766          !
6768          IF C_cnt=0 THEN ! selection process aborted
6770            End_search=1
6772            MAT Fls$= ("")
6774          ELSE
6776            MAT SORT Choose(*)
6778            FOR I=1 TO C_cnt
6780              IF Bd$(Choose(I))[18,22]=Ftype$ THEN
6782                Fls$(I)=Bd$(Choose(I))[1;14]
6784                End_search=1
6786              ELSE  ! it must be a Directory or message.
6788                SELECT Bd$(Choose(I))[18,22]
6790                CASE "up ON" ! move up one directory
6792                  LOOP
6794                    Ms_path$=Ms_path$[1,LEN(Ms_path$)-1]
6796                  EXIT IF LEN(Ms_path$)=0
6798                    Test$=Ms_path$[LEN(Ms_path$);1]
6800                  EXIT IF Test$="/"
6802                  END LOOP
6804                CASE "ROOT "  ! jump to root directory
6806                  Ms_path$=""
6808                CASE "DIR  "  ! add directory to Ms_path$
6810                  Test$=TRIM$(Bd$(Choose(I))[1,14])
6812                  Ms_path$=Ms_path$&Test$&"/"
6814                CASE ELSE
6816                  DISP "ERROR in directory jump"
6818                  PAUSE
6820                END SELECT
```

```
6822                        I=C_cnt
6824                      END IF
6826                    NEXT I
6828                  END IF
6830              END IF
6832            ELSE
6834              DISP " This directory contains no ";Ftype$;" files ... "
6836              WAIT 2.5
6838              End_search=1
6840            END IF
6842            DISP CHR$(12)
6844        UNTIL End_search
6846        SUBEXIT
6848 Cat_errors:!
6850        DISP " ERROR ... ";ERRM$
6852        BEEP
6854        CALL Pause_key_on
6856        DISP CHR$(12)
6858        C_cnt=0
6860        MAT Fls$= ("")
6862        SUBEXIT
6864        !
6866        ! ////////////////////////////////////////////////
6868        !
6870 Read_data_id: ! This routine expects to see Ids$ from
6872              ! GRAPH_DATA raw data files.
6874        DISP " Reading file contents ... Please stand by. "
6876        PRINT TABXY(1,18);" Reading #";
6878        FOR I=1 TO Bd_cnt    ! each BDAT file
6880          PRINT TABXY(11,18);
6882          PRINT USING "3D,4A,3D,2A,#";I," of ",Bd_cnt,". "
6884          Ids$="Data not recognized."
6886          IF Bd$(I)[18,22]="BDAT " THEN
6888            ON ERROR GOTO Not_recognized
6890            ASSIGN @Io_path TO Bd$(I)[1;14]
6892            ENTER @Io_path;Stat$
6894            SELECT Stat$
6896            CASE "N"
6898              ENTER @Io_path;Ids$
6900            CASE "Y"
6902              Ids$="Complete graph in GRAPH_DATA form."
6904            END SELECT
6906 Not_recognized:ASSIGN @Io_path TO *
6908            OFF ERROR
6910            IF Dir_on=2 THEN
6912              GOSUB Interpret_1
6914              IF Format_error THEN GOTO Other_format
6916              GOTO Go_on
6918            END IF
6920 Other_format:!
6922            Bd$(I)[23,71]=" ... "&Ids$
6924          END IF
6926 Go_on:NEXT I
6928        PRINT TABXY(1,18);RPT$(" ",40);
6930        DISP CHR$(12);
6932        RETURN
```

```
6934      !
6936      ! ////////////////////////////////////////////////
6938      !
6940 Interpret_1: ! This is used to interpret ID strings.
6942      Format_error=1
6944      ! identify this particular format
6946      RETURN
6948      !
6950      ! ////////////////////////////////////////////////
6952      !
6954 List_directory: ! This routine will provide a tabular listing of
6956              ! the directory along with lds$ if provided
6958              !
6960      DISP " Listing directory ... "
6962      ON TIMEOUT 7,10 GOTO Printer_kaput
6964      PRINTER IS Printer
6966      PRINT USING "//"
6968      PRINT T$
6970      IF LEN(Ms_path$)>0 THEN PRINT "HFS Path:  ";Ms_path$
6972      PRINT RPT$("=",80)
6974      PRINT "File name";
6976      IF Dir_on THEN
6978         PRINT "     - TYPE  ... contents"
6980      ELSE
6982         PRINT "     - TYPE"
6984      END IF
6986      PRINT RPT$("-",80)
6988      FOR I=1 TO Bd_cnt
6990         IF Bd$(I)[18,22]=Ftype$ OR Bd$(I)[18,22]="DIR  " THEN
6992            PRINT Bd$(I)
6994         END IF
6996      NEXT I
6998      PRINT RPT$("_",80)
7000      PRINT
7002      PRINTER IS CRT
7004      OFF TIMEOUT 7
7006      RETURN
7008 Printer_kaput:DISP " Printer not responding ... listing aborted. "
7010      BEEP
7012      WAIT 1.8
7014      OFF TIMEOUT 7
7016      RETURN
7018   SUBEND
7020   !
7022   ! ************************************************************
7024   !
7026   SUB Menu_scroll(D$,T$,Items$(*),INTEGER Item_cnt,To_select,Choose(*))
7028 Menu_scroll:! Original: 22 Jun 1987, Galen Koepke, NBS 723.04
7030           ! Revision: 22 Aug 1990, 12:00, Dennis Camell
7032           !
7034           ! A general purpose menu utility for scrolling items and
7036           ! selecting either a fixed number or a random number
7038           ! of items.
7040           !  for fixed  : To_select > 0
7042           !  for random : To_select = -1
7044           ! The items are arranged in screens of 15 items each and
```

```
7046          ! the user may access screens via softkeys. There may be
7048          ! up to 40 screens or 600 items to choose from.
7050          ! Maximum sizes: D$[80], T$[51], Items(*)[70]
7052          ! Items$(*) contains the item descriptions
7054          ! Item_cnt is the number of items in Items$(*)
7056          ! Choose(*) is dimensioned to the number of required choices
7058          !          and will be filled with the item numbers chosen.
7060          ! To_select is the number of required choices.
7062          !
7064       OPTION BASE 1
7066       PRINTER IS CRT
7068       DEG
7070       GOSUB Def_variables
7072       GOSUB Define_screens
7074       GOSUB Make_selections
7076       IF Null_file THEN  ! reset to zero
7078          Item_cnt=0
7080          Items$(1)=""
7082          To_select=0    ! no valid selections
7084       END IF
7086       SUBEXIT
7088       !
7090       ! ////////////////////////////////////////////////
7092       !
7094 Def_variables:!
7096       COM /Interrupts/ INTEGER Intr_prty
7098       COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
7100       COM /Sys/ Sys_id$[10]
7102       !
7104       INTEGER Screen_cnt,Items_per_scn,First_item(40),Last_item(40)
7106       INTEGER I,J,K,First_line,Last_line,Active_screen,Pointer,Last_pt
7108       INTEGER Local_prty,Skips,Knobcount,Pointeractive,K0,Null_file
7110       INTEGER Exit_flag,Temp,Random_select,Indx
7112       DIM Marker$[8],Test$[256]
7114       !
7116       ! initialize parameters
7118       !
7120       Local_prty=Intr_prty
7122       IF Local_prty<1 THEN Local_prty=10
7124       IF LEN(Sys_id$)=0 THEN Sys_id$=SYSTEM$("SYSTEM ID")
7126       IF Item_cnt<1 THEN
7128          Null_file=1
7130          Item_cnt=1
7132          To_select=0
7134          Items$(1)="**** Empty ****"
7136       ELSE
7138          Null_file=0
7140       END IF
7142       IF To_select=-1 THEN
7144          Random_select=1   ! choose random number of items
7146          To_select=0       ! needed for softkeys
7148       END IF
7150       IF To_select>Item_cnt THEN To_select=Item_cnt
7152       MAT Choose= (999)
7154       Skips=0
7156       Knobcount=0
```

```
7158      Doneflag=0
7160      Marker$="====>"&RPT$(CHR$(8),4)
7162      RETURN
7164      !
7166      ! ////////////////////////////////////////////////
7168      !
7170 Define_screens:! Set up screens of 15 items each.
7172            !
7174      Items_per_scn=15    ! Maximum number of displayable items
7176      IF INT(Item_cnt/Items_per_scn)=Item_cnt/Items_per_scn  THEN
7178         Screen_cnt=INT(Item_cnt/Items_per_scn)
7180      ELSE
7182         Screen_cnt=INT(Item_cnt/Items_per_scn)+1
7184      END IF
7186      J=1
7188      FOR I=1 TO Screen_cnt   ! set up each screen
7190         First_item(I)=J
7192         IF J+Items_per_scn-1<Item_cnt THEN
7194            Last_item(I)=J+Items_per_scn-1
7196            J=J+Items_per_scn
7198         ELSE
7200            Last_item(I)=Item_cnt
7202         END IF
7204      NEXT I
7206      RETURN
7208      !
7210      ! ////////////////////////////////////////////////
7212      !
7214 Make_selections:! MENU setup and use.
7216      Active_screen=1   ! first screen is active
7218      First_line=2      ! first printed line on screen  =  2 or greater.
7220      GOSUB Write_screen  ! activate screen at Active_screen
7222                          ! and set First_line and Last_line for Pointer
7224                          ! write Marker$ to first non-selected line.
7226      K0=0                ! Keys start at zero
7228      Exit_flag=0         ! allow ENTER key to exit when selections filled.
7230 Key_loop:  !
7232      ON KBD,Local_prty GOSUB Process_kbd
7234      ON KNOB .01,Local_prty GOSUB Move_pointer
7236      IF Random_select THEN
7238         ! set keys for random selection
7240         DISP D$
7242         ON KEY K0 LABEL " Select",Local_prty GOSUB Select_random
7244         ON KEY K0+9 LABEL " Accept",Local_prty GOTO Exit_line
7246      ELSE ! set key K0 for fixed selection
7248         IF Skips<To_select THEN
7250            DISP D$
7252            IF To_select>1 THEN
7254               Test$=" Select "&VAL$(Skips+1)&" of "&VAL$(To_select)
7256            ELSE
7258               Test$=" Select"
7260            END IF
7262            ON KEY K0 LABEL Test$,Local_prty GOSUB Select_fixed
7264         ELSE
7266            IF To_select>0 THEN
7268               DISP " Selection process complete ..."
```

```
7270            ELSE
7272               DISP " Menu for information only ... "
7274            END IF
7276            ON KEY K0 LABEL "Accept",Local_prty GOTO Exit_line
7278         END IF
7280      END IF
7282      IF Active_screen<Screen_cnt THEN
7284         ON KEY K0+1 LABEL " Next   Screen",Local_prty GOSUB Next_screen
7286      ELSE
7288         OFF KEY K0+1
7290      END IF
7292      IF Active_screen>1 THEN
7294         ON KEY K0+2 LABEL " Last   Screen",Local_prty GOSUB Last_screen
7296      ELSE
7298         OFF KEY K0+2
7300      END IF
7302      IF Skips>0 OR Random_select THEN
7304         ON KEY K0+3 LABEL " Reset  Select",Local_prty GOSUB Select_reset
7306      ELSE
7308         OFF KEY K0+3
7310      END IF
7312      IF To_select>0 OR Random_select THEN
7314         ON KEY K0+4 LABEL " Abort ",Local_prty GOTO Escape_line
7316      ELSE
7318         OFF KEY K0+4
7320      END IF
7322      IF Screen_cnt>2 THEN
7324         ON KEY K0+6 LABEL "Jump to Screen",Local_prty GOSUB Jump_to_scn
7326      ELSE
7328         OFF KEY K0+6
7330      END IF
7332      IF Exit_flag THEN Exit_line
7334      GOTO Key_loop
7336 Escape_line:Skips=0
7338      MAT Choose= (0)
7340      To_select=0
7342 Exit_line:OFF KEY
7344      MAT SORT Choose(*)
7346      OFF KNOB
7348      OFF KBD
7350      OUTPUT KBD;CHR$(255)&CHR$(75);
7352      PRINT CHR$(128);
7354      ! everything cleared, now go back to work.
7356      RETURN
7358      !
7360      ! ////////////////////////////////////////////////
7362      !
7364 Next_screen:     !
7366      OFF KBD
7368      OFF KNOB
7370      OFF KEY
7372      IF Active_screen=Screen_cnt THEN RETURN
7374      Active_screen=Active_screen+1
7376      GOSUB Write_screen
7378      RETURN
7380      !
```

```
7382      ! ////////////////////////////////////////////////
7384      !
7386 Last_screen:      !
7388      OFF KBD
7390      OFF KNOB
7392      OFF KEY
7394      IF Active_screen=1 THEN RETURN
7396      Active_screen=Active_screen-1
7398      GOSUB Write_screen
7400      RETURN
7402      !
7404      ! ////////////////////////////////////////////////
7406      !
7408 Jump_to_errors:DISP " Not a valid screen number ... try again. "
7410      BEEP
7412      WAIT 1.8
7414 Jump_to_scn: !
7416      OFF KBD
7418      OFF KNOB
7420      OFF KEY
7422      DISP " ENTER the screen number desired (1 to ";Screen_cnt;").";
7424      LINPUT Test$
7426      Test$=TRIM$(Test$)
7428      IF LEN(Test$)=0 THEN Jump_to_return
7430      ON ERROR GOTO Jump_to_errors
7432      Temp=INT(VAL(Test$))
7434      OFF ERROR
7436      IF Temp<1 OR Temp>Screen_cnt THEN Jump_to_errors
7438      Active_screen=Temp
7440      GOSUB Write_screen
7442 Jump_to_return: !
7444      DISP CHR$(12)
7446      Test$=""
7448      RETURN
7450      !
7452      ! ////////////////////////////////////////////////
7454      !
7456 Select_fixed:!
7458      OFF KBD
7460      OFF KNOB
7462      OFF KEY
7464      IF NOT Pointeractive THEN
7466         DISP "NO additional selections for this screen."
7468         BEEP
7470         WAIT 2
7472         DISP CHR$(12);
7474         RETURN
7476      END IF
7478      IF Skips=To_select THEN
7480         IF To_select=0 THEN
7482            DISP "This menu is for information only,";
7484            DISP " no selection allowed."
7486         ELSE
7488            DISP "All selections have been filled,";
7490            DISP " 'Select Reset' to repeat."
7492         END IF
```

```
7494        BEEP
7496        WAIT 2
7498        DISP CHR$(12);
7500        RETURN
7502     END IF
7504     Skips=Skips+1
7506     Choose(Skips)=First_item(Active_screen)+Pointer-First_line
7508     PRINT CHR$(129);  ! inverse video
7510     PRINT TABXY(10,Pointer);Items$(Choose(Skips))
7512     PRINT CHR$(128);
7514     PRINT TABXY(1,Pointer);
7516     SELECT Pointer
7518     CASE First_line
7520        GOSUB Point_forward
7522     CASE Last_line
7524        GOSUB Point_backward
7526     CASE ELSE
7528        ! move forward unless it requires wrapping to beginning.
7530        IF Skips-1>0 THEN  ! check for selected items.
7532           I=Pointer-First_line
7534           LOOP
7536              K=0
7538              FOR J=1 TO Skips
7540                 IF First_item(Active_screen)+I=Choose(J)  THEN K=1
7542              NEXT J
7544           EXIT IF K=0
7546              I=I+1
7548              IF I+First_line>Last_line THEN K=-1
7550           EXIT IF K=-1
7552           END LOOP
7554           IF K=0 THEN
7556              GOSUB Point_forward
7558           ELSE
7560              GOSUB Point_backward
7562           END IF
7564        ELSE
7566           GOSUB Point_forward
7568        END IF
7570     END SELECT
7572     RETURN
7574     !
7576     ! ////////////////////////////////////////////////
7578     !
7580 Select_random:!
7582     OFF KBD
7584     OFF KNOB
7586     OFF KEY
7588     Test$="NO"
7590     IF NOT Pointeractive THEN
7592        DISP "NO additional selections for this screen."
7594        BEEP
7596        WAIT 2
7598        DISP CHR$(12);
7600        RETURN
7602     END IF
7604     FOR I=1 TO To_select
```

```
7606        IF Choose(I) = First_item(Active_screen) + Pointer-First_line  THEN
7608           Indx = I
7610           Test$ = "YES"
7612        END IF
7614     NEXT I
7616     SELECT Test$
7618     CASE "YES"              ! Selected item is tagged ... untag
7620        IF Pointer < > Last_item(Active_screen) + 1  AND Pointer < > 17 THEN
7622           PRINT CHR$(128);! normal video
7624        ELSE
7626           PRINT CHR$(132);! underline video
7628        END IF
7630        PRINT TABXY(10,Pointer);Items$(Choose(Indx))
7632        FOR I = Indx TO To_select-1
7634           Choose(I) = Choose(I + 1)
7636        NEXT I
7638        Choose(To_select) = 999
7640        To_select = To_select-1
7642     CASE "NO"               ! Selected item is untagged ... tag it
7644        To_select = To_select + 1
7646        Choose(To_select) = First_item(Active_screen) + Pointer-First_line
7648        IF Pointer < > Last_item(Active_screen) + 1  AND Pointer < > 17 THEN
7650           PRINT CHR$(129);! inverse video
7652        ELSE
7654           PRINT CHR$(133);! inverse video with underline
7656        END IF
7658        PRINT TABXY(10,Pointer);Items$(Choose(To_select))
7660     END SELECT
7662     PRINT CHR$(128);
7664     PRINT TABXY(1,Pointer);
7666     RETURN
7668     !
7670     ! ////////////////////////////////////////////////
7672     !
7674 Select_reset:     !Clear Choose file
7676     OFF KBD
7678     OFF KNOB
7680     OFF KEY
7682     IF Random_select THEN To_select = 0
7684     Skips = 0
7686     MAT Choose = (999)
7688     GOSUB Write_screen
7690     RETURN
7692     !
7694     ! ////////////////////////////////////////////////
7696     !
7698 Process_kbd:! Allow use of arrows and enter key in addition to soft.
7700     Test$ = KBD$
7702     IF LEN(Test$) = 1 AND Test$[1,1] < > CHR$(32) THEN
7704        BEEP 80.,.1
7706        RETURN
7708     END IF
7710     IF Test$[1,1] = CHR$(32) THEN GOSUB Point_forward
7712     IF Test$[1,1] < > CHR$(255) THEN RETURN
7714     SELECT Test$[2,2]
7716     CASE CHR$(255)
```

```
7718              ! do nothing
7720      CASE "V","T"
7722         GOSUB Point_forward
7724      CASE "^","W"
7726         GOSUB Point_backward
7728      CASE "E","s","t","&"
7730         IF Random_select THEN
7732            GOSUB Select_random
7734         ELSE
7736            IF Skips<To_select THEN
7738               GOSUB Select_fixed
7740            ELSE
7742            ! exit routine
7744               Exit_flag=1
7746            END IF
7748         END IF
7750      CASE ELSE
7752         BEEP 80.,.1
7754      END SELECT
7756      Test$=""
7758      RETURN
7760      !
7762      ! ////////////////////////////////////////////////
7764      !
7766 Point_forward:Knobcount=5
7768      GOSUB Move_pointer
7770      RETURN
7772 Point_backward:Knobcount=-5
7774      GOSUB Move_pointer
7776      RETURN
7778      !
7780      ! ////////////////////////////////////////////////
7782      !
7784 Jog_pointer:! Move the selection pointer on the active screen.
7786           ! without regard to selected values
7788      IF Knobcount>0 THEN  ! Move forward
7790         Pointer=Pointer+1
7792      ELSE                 ! Move backward
7794         Pointer=Pointer-1
7796      END IF
7798      IF Pointer<First_line THEN Pointer=Last_line
7800      IF Pointer>Last_line THEN Pointer=First_line
7802      RETURN
7804      !
7806      ! ////////////////////////////////////////////////////
7808      !
7810 Move_pointer:! Control pointer to avoid re-selection of items
7812      IF NOT Pointeractive THEN RETURN  ! No selections to be made.
7814      Knobcount=Knobcount+KNOBX-KNOBY
7816      IF ABS(Knobcount)<4 THEN RETURN
7818      Last_pt=Pointer
7820      GOSUB Jog_pointer
7822      IF Skips>0 THEN
7824         LOOP
7826            J=Pointer-First_line
7828            FOR I=1 TO Skips
```

```
7830              IF First_item(Active_screen)+J=Choose(I) THEN J=999
7832            NEXT I
7834            IF J=999 AND Pointer=Last_pt THEN Pointeractive=0
7836          EXIT IF Pointeractive=0
7838            IF J=999 THEN GOSUB Jog_pointer
7840          EXIT IF J<>999
7842          END LOOP
7844        END IF
7846        Knobcount=0
7848        OUTPUT KBD;CHR$(255)&CHR$(84);  ! Bring screen home
7850        IF Last_pt=Last_line THEN PRINT CHR$(132);
7852        PRINT "    ";
7854        IF Pointeractive THEN ! Pointer active
7856          IF Pointer=Last_line THEN
7858            PRINT CHR$(132);
7860          ELSE
7862            PRINT CHR$(128);
7864          END IF
7866          PRINT TABXY(1,Pointer);Marker$;CHR$(128);
7868        END IF
7870        RETURN
7872        !
7874        ! ////////////////////////////////////////////////
7876        !
7878 Write_screen:! Write the screen pointed to by Active_screen
7880        ! home and clear screen
7882        OUTPUT KBD;CHR$(255)&CHR$(84)&CHR$(255)&CHR$(75);
7884        Knobcount=KNOBX-KNOBY   ! Clear knob and keyboard
7886        Knobcount=0
7888        Test$=KBD$
7890        Test$=""
7892        !
7894        PRINT TABXY(1,First_line-1);CHR$(132);" Item #| Screen #";
7896        PRINT USING "#,2D,4A,2D,3A";Active_screen," of ";Screen_cnt;" | "
7898        PRINT T$;RPT$(" ",51-LEN(T$));
7900        PRINT TABXY(80,First_line-1);"|";CHR$(128);
7902        J=0
7904        REPEAT
7906          IF J=Last_item(Active_screen)-First_item(Active_screen) THEN
7908            PRINT CHR$(132);
7910            PRINT TABXY(1,First_line+J);RPT$(" ",80)
7912          ELSE
7914            PRINT CHR$(128);
7916          END IF
7918          PRINT TABXY(5,First_line+J);
7920          PRINT USING "3D,A,#";First_item(Active_screen)+J,"|"
7922          IF Random_select THEN
7924            FOR I=1 TO To_select
7926              IF First_item(Active_screen)+J=Choose(I) THEN
7928                PRINT CHR$(129);
7930              END IF
7932            NEXT I
7934          ELSE
7936            IF Skips>0 THEN ! make this line inverse video
7938              FOR I=1 TO Skips
7940                IF First_item(Active_screen)+J=Choose(I) THEN
```

```
7942                    PRINT CHR$(129);
7944                  END IF
7946                NEXT I
7948              END IF
7950            END IF
7952            PRINT TABXY(10,First_line+J);Items$(First_item(Active_screen)+J)
7954            PRINT TABXY(80,First_line+J);"|";
7956            J=J+1
7958        UNTIL J>=(Last_item(Active_screen)-First_item(Active_screen)+1)
7960        Last_line=Last_item(Active_screen)-First_item(Active_screen)
7962        Last_line=Last_line+First_line
7964        !
7966        ! set marker to first non-selected item.
7968        !
7970        Pointeractive=0
7972        IF To_select>0 OR Random_select THEN Pointeractive=1
7974        IF Skips>0 AND Pointeractive=1 THEN ! find first non-selected item
7976            J=0
7978            LOOP
7980                Pointer=First_line+J
7982                FOR I=1 TO Skips
7984                    IF First_item(Active_screen)+J=Choose(I) THEN Pointer=0
7986                NEXT I
7988            EXIT IF Pointer<>0
7990                J=J+1
7992                IF First_line+J>Last_line THEN
7994                    Pointeractive=0
7996                    Pointer=First_line
7998                END IF
8000            EXIT IF Pointer<>0
8002            END LOOP
8004        ELSE
8006            Pointer=First_line
8008        END IF
8010        IF Pointeractive THEN
8012            IF Pointer=Last_line THEN
8014                PRINT CHR$(132);
8016            ELSE
8018                PRINT CHR$(128);
8020            END IF
8022            PRINT TABXY(1,Pointer);Marker$;CHR$(128);
8024        END IF
8026        RETURN
8028    SUBEND
8030    !
8032    ! ************************************************************
8034    !
8036    SUB Errortrap
8038 Errortrap: ! Original: 13 Nov 1984
8040            ! Revision: 02 Dec 1987
8042        ! Trap most errors here
8044        OPTION BASE 1
8046        COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
8048        DIM File$[20],Test$[256],What$[20],Ac$[5]
8050        BEEP 400,.6
8052        SELECT ERRN
```

```
8054      CASE 54
8056        DISP "DUPLICATE FILE NAME:  ";Filename$;
8058        DISP "....PURGE old one? (Y/N)";
8060        LINPUT What$
8062        What$=TRIM$(What$)
8064        SELECT What$[1,1]
8066        CASE "Y","y"
8068          PURGE Ms_path$&Filename$&Diskdrive$
8070        CASE ELSE
8072          Ac$="VALID"
8074          CALL Enterfilename(Ac$)
8076        END SELECT
8078      CASE 52,53
8080        DISP "Improper FILE NAME -- ENTER NEW FILE NAME";
8082        OUTPUT 2 USING "#,K,K";"#";Filename$
8084        LINPUT Filename$
8086        Filename$=TRIM$(Filename$)
8088      CASE 56
8090        DISP "FILE:  ";Filename$;" is not on this disk, please insert";
8092        DISP " correct disk"
8094        CALL Pause_key_on
8096      CASE 64
8098        DISP "This disk is full, PLEASE insert clean disk"
8100        CALL Pause_key_on
8102      CASE 56
8104        DISP "DATA INPUT disk must be in drive!!  ";
8106        DISP "...CONTINUE when ready."
8108        CALL Pause_key_on
8110      CASE 72,73,76
8112        DISP Diskdrive$;
8114        DISP " is not available, type correct";
8116        DISP " unit specifier (ie. ':,707,0')."
8118        OUTPUT 2 USING "K,#";Diskdrive$
8120        LINPUT Diskdrive$
8122      CASE 80
8124        DISP "CHECK DISK drive door!"
8126        CALL Pause_key_on
8128      CASE ELSE
8130        DISP ERRM$;" 'CONTINUE' when fixed"
8132        CALL Pause_key_on
8134      END SELECT
8136      DISP CHR$(12)
8138      SUBEXIT
8140    SUBEND
8142    !
8144    ! ************************************************************
8146    !
```

```
7942                    PRINT CHR$(129);
7944                  END IF
7946                NEXT I
7948              END IF
7950            END IF
7952            PRINT TABXY(10,First_line+J);Items$(First_item(Active_screen)+J)
7954            PRINT TABXY(80,First_line+J);"|";
7956            J=J+1
7958        UNTIL J>=(Last_item(Active_screen)-First_item(Active_screen)+1)
7960        Last_line=Last_item(Active_screen)-First_item(Active_screen)
7962        Last_line=Last_line+First_line
7964        !
7966        ! set marker to first non-selected item.
7968        !
7970        Pointeractive=0
7972        IF To_select>0 OR Random_select THEN Pointeractive=1
7974        IF Skips>0 AND Pointeractive=1 THEN ! find first non-selected item
7976            J=0
7978            LOOP
7980                Pointer=First_line+J
7982                FOR I=1 TO Skips
7984                    IF First_item(Active_screen)+J=Choose(I) THEN Pointer=0
7986                NEXT I
7988            EXIT IF Pointer<>0
7990                J=J+1
7992                IF First_line+J>Last_line THEN
7994                    Pointeractive=0
7996                    Pointer=First_line
7998                END IF
8000            EXIT IF Pointer<>0
8002            END LOOP
8004        ELSE
8006            Pointer=First_line
8008        END IF
8010        IF Pointeractive THEN
8012            IF Pointer=Last_line THEN
8014                PRINT CHR$(132);
8016            ELSE
8018                PRINT CHR$(128);
8020            END IF
8022            PRINT TABXY(1,Pointer);Marker$;CHR$(128);
8024        END IF
8026        RETURN
8028    SUBEND
8030    !
8032    ! ************************************************************
8034    !
8036    SUB Errortrap
8038 Errortrap: ! Original: 13 Nov 1984
8040            ! Revision: 02 Dec 1987
8042        ! Trap most errors here
8044        OPTION BASE 1
8046        COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
8048        DIM File$[20],Test$[256],What$[20],Ac$[5]
8050        BEEP 400,.6
8052        SELECT ERRN
```

```
8054      CASE 54
8056        DISP "DUPLICATE FILE NAME:  ";Filename$;
8058        DISP "....PURGE old one? (Y/N)";
8060        LINPUT What$
8062        What$=TRIM$(What$)
8064        SELECT What$[1,1]
8066        CASE "Y","y"
8068          PURGE Ms_path$&Filename$&Diskdrive$
8070        CASE ELSE
8072          Ac$="VALID"
8074          CALL Enterfilename(Ac$)
8076        END SELECT
8078      CASE 52,53
8080        DISP "Improper FILE NAME -- ENTER NEW FILE NAME";
8082        OUTPUT 2 USING "#,K,K";"#";Filename$
8084        LINPUT Filename$
8086        Filename$=TRIM$(Filename$)
8088      CASE 56
8090        DISP "FILE:  ";Filename$;" is not on this disk, please insert";
8092        DISP " correct disk"
8094        CALL Pause_key_on
8096      CASE 64
8098        DISP "This disk is full, PLEASE insert clean disk"
8100        CALL Pause_key_on
8102      CASE 56
8104        DISP "DATA INPUT disk must be in drive!!  ";
8106        DISP "...CONTINUE when ready."
8108        CALL Pause_key_on
8110      CASE 72,73,76
8112        DISP Diskdrive$;
8114        DISP " is not available, type correct";
8116        DISP " unit specifier (ie. ':,707,0')."
8118        OUTPUT 2 USING "K,#";Diskdrive$
8120        LINPUT Diskdrive$
8122      CASE 80
8124        DISP "CHECK DISK drive door!"
8126        CALL Pause_key_on
8128      CASE ELSE
8130        DISP ERRM$;" 'CONTINUE' when fixed"
8132        CALL Pause_key_on
8134      END SELECT
8136      DISP CHR$(12)
8138      SUBEXIT
8140    SUBEND
8142    !
8144    ! ************************************************************
8146    !
```

```
100  ! RE-STORE "DECON_NIST:,702"
102  !
104   COM /Sys/ Sys_id$[10]
106   COM /Sys_msi/ Msi_id$[20]
108  !
110   OUTPUT KBD USING "K,#";"SCRATCH KEYE"        !ERASE SOFT KEYS
112   CONTROL KBD,15;0! sets the color of the soft keys
114   CONTROL KBD,2;1
116  !
118   Intr_prty=1
120   CALL Decon
122  !
124   CLEAR SCREEN
126   OUTPUT KBD USING "K,#";"LOAD KEYE"! restore the typing aid keys
128   PRINT TABXY(1,5);"END of program. So long."
130   !
132   !*************************************************************************
134   !
136   ! Written by S.M. Chesnut at the National Institute of Standards
138   ! and Technology.
140   !
142   !*********************************
144 Date_line:    ! April 10, 1991
146   !*********************************
148  !
150   END
152  !
154  !_____________________________________________________________
156  !
158   SUB Decon
160   !
162      OPTION BASE 1
164      COM /Interrupts/ INTEGER Intr_prty
166      COM /Waveforms/ COMPLEX Resp(4096),COMPLEX Wave(4096),COMPLEX Reg(4096)
168      COM /Waveforms/ REAL Dif_op(4096),COMPLEX Decn(4096),COMPLEX Td(4096)
170      COM /Data_stuff/ INTEGER Number,REAL Delta_x,REAL Strt_time
172      COM /Deconv/ REAL Gamma,INTEGER Min_flag
174   !
176 Decon:   !
178   !
180      !
182      DIM Ch$[1],Data_id$[40],Set_dc$[1]
184      INTEGER Itemp,Flg,Extended,Resp_ext,Typ,No_fft
186      REAL Rtemp,Sum,Dum,Atten
188      COMPLEX C_temp
190      COMPLEX Sigma,W
192      !
194      RAD
196      OFF KEY
198      CLEAR SCREEN
200      Ch$=""
202      Number=32767
204      Resp_ext=0
206      Extended=0
208      Typ=0
```

```
210     No_fft=0
212     CALL Get_data(Wave(*),Dum,Flg)
214     Rtemp=Delta_x
216     INPUT "Nahaman-Gans extend the waveform? y/n (default is n)",Ch$
218     IF (Ch$="Y") OR (Ch$="y") THEN
220        Ch$=""
222        INPUT "Is the waveform impulse-like? y/n (default is n)",Ch$
224        IF (Ch$="y") OR (Ch$="Y") THEN Typ=1
226        CALL Ng_extend(Wave(*),Typ)
228        Extended=1
230     END IF
232     Itemp=Number
234     Typ=0
236     REDIM Wave(Number)
238     CALL Get_data(Resp(*),Sum,Flg)
240     Int_resp=(Sum-.5*(Resp(1)+Resp(Number)))*Delta_x
242     INPUT "Nahaman-Gans extend the waveform? y/n (default is n)",Ch$
244     IF (Ch$="Y") OR (Ch$="y") THEN
246        Ch$=""
248        INPUT "Is the waveform impulse-like y/n (default is n)",Ch$
250        IF (Ch$="y") OR (Ch$="Y") THEN Typ=1
252        CALL Ng_extend(Resp(*),Typ)
254        Resp_ext=1
256     END IF
258     REDIM Resp(Number)
260     !
262     Bug1=0
264     IF Bug1 THEN
266        PRINT Number,Itemp
268        PRINT Delta_x,Rtemp
270        PRINT "Type continue to go on."
272        PAUSE
274     END IF
276     !
278     IF (Itemp<>Number) OR (DROUND(Rtemp,3)<>DROUND(Delta_x,3)) THEN
280        BEEP
282        DISP "The system and the output waveforms are inconsistent."
284        WAIT 1.0
286        DISP "This program is ended."
288        SUBEXIT
290     END IF
292     !
294     Set_dc$="y"
296     INPUT "Set the dc level ? y/n (default = y)",Set_dc$
298     !
300     Ch$="y"
302     INPUT "Do you want iterative deconvolution? y/n (default = y)",Ch$
304     !
306     !
308     IF (Ch$="N") OR (Ch$="n") THEN
310        INPUT "What is the value for gamma?",Gamma
312        CALL Do_fft(Resp(*),1,No_fft)
314        IF No_fft THEN SUBEXIT
316        CALL Do_fft(Wave(*),1,No_fft)
318        IF No_fft THEN SUBEXIT
320        CALL Gam(Sum,Imag_sum,Set_dc$)
322     ELSE
324        CALL Do_fft(Wave(*),1,No_fft)
```

```
326         IF No_fft THEN SUBEXIT
328         CALL Do_fft(Resp(*),1,No_fft)
330         IF No_fft THEN SUBEXIT
332         CALL Iterate(Sum,Set_dc$)
334       END IF
336       !
338       IF (Extended) AND (NOT Typ) THEN
340         MAT Td= (2)*Td
342       END IF
344       CALL Record_data(Extended,Sum)
346       SUBEXIT
348     SUBEND
350     !
352     !*************************************************************************
354     !
356     SUB Ng_extend(COMPLEX Wave(*),INTEGER Typ)
358     !
360   Ng_extend:   !
362     ! This routine performs the Nahman-Gans waveform extension for step-
364     ! like and square-like waveforms, and "pads" impulse-like waveforms
366     ! with zeros.
368     ! The array 'waveform' is a complex data array which contains the
370     ! data to be extended.
372     !
374     ! Span is the sum of the beginning and ending values of the
376     ! waveform.
378     !
380       COM /Data_stuff/ INTEGER Number,REAL Delta_x,REAL Strt_time
382     !
384     !
386       DIM Ch$[1]
388     !
390       IF Typ THEN
392   Try_again:      !
394         INPUT "Extend to a) 2048 or b) 4096 points. Enter letter.",Ch$
396         SELECT Ch$
398         CASE "a","A"
400           Ndex=2048
402         CASE "b","B"
404           Ndex=4096
406         CASE ELSE
408           BEEP
410           DISP "Error in selection, try again."
412           WAIT 1.0
414           GOTO Try_again
416         END SELECT
418       END IF
420       IF Typ<>1 THEN
422         FOR I=1 TO Number
424           Ndex=I+Number
426           Wave(Ndex)=Wave(Number)+Wave(1)-Wave(I)
428         NEXT I
430         Number=Number*2
432       ELSE
434         FOR I=Number+1 TO Ndex
436           Wave(I)=CMPLX(0,0)
```

```
438         NEXT I
440         Number=Ndex
442       END IF
444   ! Extension complete.
446       SUBEXIT
448   SUBEND
450   !
452   !*************************************************************************
454   !
456   SUB Iterate(REAL Sum,Set_dc$)
458   !
460 Iterate:   !
462         !
464   ! This routine sets up the iterative deconvolution.
466   ! The actual deconvolution is performed in the sub GAMMA.
468   !
470       OPTION BASE 1
472       COM /Interrupts/ INTEGER Intr_prty
474       COM /Waveforms/ COMPLEX Resp(4096),COMPLEX Wave(4096),COMPLEX Reg(4096)
476       COM /Waveforms/ REAL Dif_op(4096),COMPLEX Decn(4096),COMPLEX Td(4096)
478       COM /Data_stuff/ INTEGER Number,REAL Delta_x,REAL Strt_time
480       COM /Deconv/ REAL Gamma,INTEGER Min_flag
482   !
484       ALLOCATE Ht_gam(1000,2)
486       REAL Step,Strt,Stop,Min_sum,Last_min,Inc
488       INTEGER I,J,Min_pos,Done,Exists
490       DIM C$[1],Data_id$[40]
492       !
494       Done=0
496       Step=1
498       INPUT "What is the starting attenuation value?",Strt
500       INPUT "What is the stopping attenuation value?",Stop
502       J=1
504       MAT Ht_gam= (0)
506       WHILE NOT Done
508         REDIM Ht_gam(1000,2)
510         FOR Inc=Strt TO Stop STEP Step
512           Gamma=10^Inc
514           Exists=0
516           MAT SEARCH Ht_gam(*,1),LOC(=Gamma);Exists
518           IF (Exists<1) OR (Exists>1000) THEN
520             CALL Gam(Sum,Imag_sum,Set_dc$)
522             Ht_gam(J,1)=Gamma
524             Ht_gam(J,2)=Imag_sum
526             J=J+1
528           END IF
530         NEXT Inc
532         REDIM Ht_gam(J-1,2)
534         MAT SORT Ht_gam(*,1)
536         GOSUB Find_min
538         IF Step<.50 THEN Done=1
540         SELECT Min_pos
542         CASE =(J-1)
544           Strt=Stop
546           Stop=Strt+2
548         CASE =1
```

```
550             Stop=Strt
552             Strt=Stop-2
554          CASE ELSE
556             Strt=LGT(Ht_gam(Min_pos,1))-Step
558             Stop=LGT(Ht_gam(Min_pos,1))+Step
560             Step=Step/4.
562          END SELECT
564       END WHILE
566       Gamma=Ht_gam(Min_pos,1)
568       CALL Gam(Sum,Ht_gam(Min_pos,2),Set_dc$)
570       J=J-1
572       INPUT "Save the imaginary sum vs gamma ? y/n",C$
574       IF (C$="y") OR (C$="Y") THEN
576          REDIM Ht_gam(J,2)
578          MAT SORT Ht_gam(*,1)
580          INPUT "Enter a 40 character or less data description.",Data_id$
582          Intr_prty=Local_prty+3
584          CALL Data_to_disk_r(Ht_gam(*),J,J,Data_id$)
586          Intr_prty=Local_prty
588       END IF
590       DEALLOCATE Ht_gam(*)
592       SUBEXIT
594 Find_min:      !
596       Bug1=0
598       IF Bug1 THEN
600          FOR I=1 TO J-1
602             PRINT Ht_gam(I,1)
604             PRINT Ht_gam(I,2)
606          NEXT I
608       END IF
610       Min_sum=Ht_gam(1,2)
612       Min_pos=1
614       FOR I=2 TO J-1
616          IF Ht_gam(I,2)<Min_sum THEN
618             Min_sum=Ht_gam(I,2)
620             Min_pos=I
622          END IF
624       NEXT I
626       RETURN
628    SUBEND
630    !
632    !*************************************************************************
634    !
636    SUB Gam(REAL Sum_resp,Img_sum,Set_dc$)
638    !
640       OPTION BASE 1
642       RAD
644       COM /Interrupts/ INTEGER Inrt_prty
646       COM /Waveforms/ COMPLEX Resp(*),COMPLEX Wave(*),COMPLEX Reg(*)
648       COM /Waveforms/ REAL Dif_op(*),COMPLEX Decn(*),COMPLEX Td(*)
650       COM /Data_stuff/ INTEGER Number,REAL Delta_x,REAL Strt_time
652       COM /Deconv/ REAL Gamma,INTEGER Min_flag
654    !
656    ! This subroutine generates the regularization filter and the
658    ! calculations for deconvolution.
660    ! NIST, Boulder, Colorado.
```

```
662   !
664       INTEGER J,No_fft
666       COMPLEX Decn_val,Ctemp
668   !
670   ! Initialize the variables.
672   !
674 Gam:  !
676       DISP "Performing the deconvolution."
678       Imag_sum=0.
680       REDIM Td(Number)
682       MAT Decn= Resp
684       IF (Set_dc$="y") OR (Set_dc$="Y") THEN
686          Decn(1)=CMPLX(ABS(Resp(2)),0.)
688       ELSE
690          IF Decn(1)=CMPLX(0.,0.) THEN Decn(1)=CMPLX(Sum_resp,0.)
692       END IF
694       Decn(1)=Wave(1)/Decn(1)
696       IF NOT Min_flag THEN Reg(1)=CMPLX(0.,0.)
698       FOR J=2 TO Number
700          IF Decn(J)=CMPLX(0.,0.) THEN
702             Decn(J)=Ctemp
704             IF NOT Min_flag THEN Reg(1)=CMPLX(0.,0.)
706          END IF
708          IF NOT Min_flag THEN
710             Magn=REAL(Resp(J))^2+IMAG(Resp(J))^2
712             Rtemp=2.*PI*(J-1)/(Number-1.)*1.
714             Dif_opr=6.0-8.0*COS(Rtemp)+2.*COS(2.*Rtemp)
716             Reg(J)=Magn/(Magn+Gamma*Dif_opr)
718          END IF
720          Ctemp=Resp(J)
722          Decn(J)=Wave(J)*Reg(J)/Resp(J)
724       NEXT J
726       MAT Td= Decn
728       Do_fft(Td(*),-1,No_fft)    !CONVERT TO TIME DOMAIN
730       Img_sum=0
732       FOR I=1 TO Number
734          Img_sum=IMAG(Td(I))^2+Img_sum
736       NEXT I
738       Img_sum=SQRT(Img_sum/Number)
740       PRINT "gamma = ",Gamma," Imaginary sum = ",Img_sum
742       PRINTER IS CRT
744       MAT Td= (1/(Delta_x*Number))*Td
746       SUBEXIT
748    SUBEND
750    !
752    !*************************************************************************
754    !
756    SUB Do_fft(COMPLEX Fft_file(*),INTEGER Fft_flg,No_fft)
758    !
760       OPTION BASE 1
762       COM /Interrupts/ INTEGER Intr_prty
764       IFFT_FIX
766       INTEGER I2
768       !
770 Do_fft: !
772    !
```

```
774       I2=0
776       DISP " Calculating an FFT ... please wait "
778       Timer=TIMEDATE
780       Intr_prty=Local_prty+3
782       CALL Fft_fix(Fft_file(*),I2,Fft_flg)
784       Intr_prty=Local_prty
786       ! Fft_file(*) returns with results, I2 is an error flag.
788       IF I2<>0 THEN
790          SELECT I2
792          CASE 1
794             DISP " Negative data count, FFT aborted."
796             WAIT 1
798          CASE 2
800             DISP " Data has zero or negative spacing, FFT aborted."
802             WAIT 1
804          CASE 3
806             DISP " Data count not a power of 2, FFT aborted."
808             WAIT 1
810          CASE ELSE
812             DISP " ERRORs in FFT, operation aborted."
814          END SELECT
816          DISP "continue ..."
818          PAUSE
820          GOTO No_fft_action
822       ELSE
824          SUBEXIT
826       END IF
828       LOOP ! This keeps the last disp (before the case stmt.) on screen
830       EXIT IF TIMEDATE-Timer>1.8
832       END LOOP
834       !
836 No_fft_action:OFF KEY
838    !
840       DISP "Due to a previously described error or another strange event, "
842       DISP "no FFT was performed."
844       No_fft=1
846       SUBEXIT
848    SUBEND
850    !
852    !*****************************************************************************
854    !
856    SUB Fft_fix(COMPLEX Fft_file(*),INTEGER Err,INTEGER Fft_flg)
858 Fft_fix:  ! Original: 04 Jul 1987,  J. Ladbury
860          ! Revision: 02 Dec 1987
862          ! Modifications: 24 Aug 90 S.M. Chesnut
864          ! As modified, this routine; checks for valid data and
866          ! splits the input array (Fft_file) into its real and imaginary
868          ! parts in preparation for the Fast Fourier Transform (FFT).
870          ! After the FFT has been performed, the real and imaginary
872          ! results are stored in the Fft_file array.
874       OPTION BASE 1
876       COM /Interrupts/ INTEGER Intr_prty
878       COM /Data_stuff/ INTEGER Number,REAL Delta_x,REAL Strt_time
880       !
882       !
884       INTEGER Local_prty,Power
```

```
886       REAL Fstep
888       !
890       Local_prty=Intr_prty
892       Err=0
894       IF Number<2 THEN ! gotta have more than one point.
896          Err=1
898          SUBEXIT
900       END IF
902       IF (Delta_x<=0) THEN ! Check for positive time.
904          Err=2
906          SUBEXIT
908       END IF
910       Fstep=1/(Delta_x*(Number))
912       Power=LOG(Number)/LOG(2)
914       IF INT((2^Power)+.5)<>Number THEN ! FFT on data which has
916                                         ! power of 2 data points.
918          Err=3
920          SUBEXIT
922       END IF
924       ALLOCATE REAL Re(Number),Im(Number)
926       MAT Re= REAL(Fft_file)
928       MAT Im= IMAG(Fft_file)
930       Intr_prty=Local_prty+3
932       Power=Power+1
934       CALL Fft_imag(Number*1.,Power*1.0,Fft_flg*1.0,Re(*),Im(*))
936       Intr_prty=Local_prty
938       MAT Fft_file= CMPLX(Re,Im)
940       Bug1=0
942       IF Bug1 THEN
944          FOR I=1 TO Number
946             PRINT Fft_file(I);I
948          NEXT I
950       END IF
952       DEALLOCATE Re(*),Im(*)
954    SUBEND
956    !
958    !*****************************************************************
960    !
962    SUB Fft_imag(N,Power,Flg,R_(*),I_(*))
964       ! Algorithm from HP library of math routines.
966       ! Modified 25 October 1990 to work on complex data.
968       ! Modification by J. Ladbury of NIST, Boulder, Colorado.
970       OPTION BASE 1
972       RAD
974       Baddta=(N<=0) OR (Flg<>1) AND (Flg<>-1) OR (Power<=0)
976       IF Baddta=0 THEN 986
978       PRINT FNLin$(2);"ERROR IN SUBPROGRAM Fft."
980       PRINT "N=";N,"Flg=";Flg,"Power=";Power;FNLin$(2)
982       CALL Pause_key_on
984       GOTO 974
986 Fft:  K=0
988       FOR J=1 TO N-1
990          I=2
992          IF K<N/I THEN 1000
994          K=K-N/I
996          I=I+I
```

```
998         GOTO 992
1000        K=K+N/I
1002        IF K<=J THEN 1016
1004        A=R_(J+1)
1006        R_(J+1)=R_(K+1)
1008        R_(K+1)=A
1010        A=I_(J+1)
1012        I_(J+1)=I_(K+1)
1014        I_(K+1)=A
1016      NEXT J
1018      G=.5
1020      P=1
1022      FOR I=1 TO Power-1
1024        G=G+G
1026        C=1
1028        E=0
1030        Q=SQR((1-P)/2)*Flg
1032        P=(1-2*(I=1))*SQR((1+P)/2)
1034        FOR R=1 TO G
1036          FOR J=R TO N STEP G+G
1038            K=J+G
1040            A=C*R_(K)+E*I_(K)
1042            B=E*R_(K)-C*I_(K)
1044            R_(K)=R_(J)-A
1046            I_(K)=I_(J)+B
1048            R_(J)=R_(J)+A
1050            I_(J)=I_(J)-B
1052          NEXT J
1054          A=E*P+C*Q
1056          C=C*P-E*Q
1058          E=A
1060        NEXT R
1062      NEXT I
1064      SUBEXIT
1066    SUBEND
1068    !
1070    ! ********************************************************
1072    !
1074    SUB Con_to_real(COMPLEX Dat(*),REAL Temp_f(*),INTEGER Flg)
1076    !
1078      OPTION BASE 1
1080    ! This subroutine converts data from COMPLEX to REAL and converts to
1082    ! the "GRAPH_DATA" format.
1084    !
1086    ! Dat() is the array which contains the complex data.
1088    ! Temp_f() is the array which contains the converted data.
1090    ! Flg indicates which part, the real (flg=0) or imaginary (flg=1)
1092    ! is being saved.
1094    ! Total is the relative time of each point.
1096    !
1098  Con_to_real:  !
1100      COM /Data_stuff/ INTEGER Number,REAL Delta_x,REAL Strt_time
1102    !
1104      REAL Total
1106    !
1108      Total=Delta_x
```

```
1110  !
1112      FOR I=1 TO Number
1114        IF NOT Flg THEN
1116          Temp_f(I,2)=REAL(Dat(I))
1118        ELSE
1120          Temp_f(I,2)=IMAG(Dat(I))
1122        END IF
1124        Temp_f(I,1)=Total
1126        Total=Total+Delta_x
1128        IF Bug2 THEN
1130          PRINT Temp_f(I,1),Temp_f(I,2)
1132          PRINT I
1134        END IF
1136      NEXT I
1138      SUBEXIT
1140  SUBEND
1142  !
1144  !*************************************************************************
1146  !
1148  SUB Get_data(COMPLEX File(*),REAL Total,INTEGER Flg)
1150  ! This converts data from x,y pair real data format to
1152  ! complex format. The first position of the data array contains
1154  ! the number of data points in the real part and the imaginary part
1156  ! contains the point spacing (delta t).
1158  ! FLG indicates a real (flg=0) data file, (flg=1) indicates a
1160  ! complex data file.
1162  ! All files are assumed to be x,y pairs by the routine load_disk_data.
1164  ! As such, it is necessary to convert every file to type COMPLEX.
1166  !
1168      OPTION BASE 1
1170      COM /Interrupts/ INTEGER Intr_prty
1172      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
1174      COM /Data_stuff/ INTEGER Number,REAL Delta_x,REAL Strt_time
1176  !
1178 Get_data:  !
1180          !
1182      REAL Temp(4096,2)
1184      INTEGER Real_or_cmplx
1186      DIM Dataid$[40]
1188      Total=0
1190 Read_again:      !
1192      Diskdrive$=""
1194      Filename$=""
1196      DISP "Enter the name of the measured waveform when prompted."
1198      WAIT .8
1200      INPUT "Is this data real = 0 or complex = 1",Flg
1202      IF (Flg<>0) AND (Flg<>1) THEN
1204        DISP "IMPROPER INPUT, please try again."
1206        WAIT 1.0
1208        GOTO Read_again
1210      END IF
1212      Real_or_cmplx=Flg
1214      Intr_prty=Local_prty+3
1216      CALL Load_disk_data(Temp(*),Number,Dataid$,Flg)
1218      Intr_prty=Local_prty-3
1220      IF Flg THEN
```

```
1222        BEEP
1224        DISP "No data was read. Please try again."
1226        WAIT 1.0
1228        GOTO Read_again
1230      END IF
1232      Intr_prty=Local_prty
1234      IF Real_or_cmplx=0 THEN
1236        FOR I=1 TO Number
1238          File(I)=CMPLX(Temp(I,2),0.)
1240          Total=Total+Temp(I,2)
1242        NEXT I
1244      ELSE
1246        FOR I=1 TO Number
1248          File(I)=CMPLX(Temp(I,1),Temp(I,2))
1250          Total=Total+ABS(File(I))
1252        NEXT I
1254      END IF
1256      Flg=Real_or_cmplx
1258      SUBEXIT
1260    SUBEND
1262    !
1264    !*****************************************************************************
1266    !
1268    SUB Record_data(INTEGER Extended,REAL Sum)
1270    !
1272  Record_data:  !
1274      OPTION BASE 1
1276      COM /Interrupts/ INTEGER Inrt_prty
1278      COM /Waveforms/ COMPLEX Resp(*),COMPLEX Wave(*),Reg(*)
1280      COM /Waveforms/ REAL Dif_op(*),COMPLEX Decn(*),COMPLEX Td(*)
1282      COM /Data_stuff/ INTEGER Number,REAL Delta_x,REAL Strt_time
1284      !
1286      DIM Data_id$[40],C$[1]
1288      COMPLEX Temp_comp(4096)
1290      INTEGER Local_prty,Num_o_pnts
1292      REAL Time_scale,Freq_scale
1294      !
1296      Local_prty=Intr_prty
1298      C$="n"
1300      Num_o_pnts=Number
1302      IF Extended THEN
1304        INPUT "Record the extended waveforms ? (default is no)",C$
1306        IF (C$<>"y") AND (C$<>"Y") THEN
1308          Num_o_pnts=Number/2
1310        END IF
1312      END IF
1314      !
1316      ALLOCATE Temp_file(4096,2)
1318      !
1320      Time_scale=Delta_x
1322      Freq_scale=1./(Delta_x*Number)
1324      !
1326      C$="n"
1328      INPUT "Save the spectrum magnitude of the deconvolved waveform ? y/n",C$
1330      IF (C$="y") OR (C$="Y") THEN
1332        INPUT "Enter a 40 character or less data description.",Data_id$
```

```
1334        Intr_prty=Local_prty+3
1336        MAT Temp_comp= Decn
1338        REDIM Temp_comp(4096)
1340        Temp=Number
1342        Delta_x=Freq_scale
1344        FOR I=2 TO Num_o_pnts
1346          IF ABS(Temp_comp(I))<>0 THEN
1348            Temp_r=2*ABS(1.E+12*Temp_comp(I))
1350            Temp_r=20*LGT(Temp_r) ! 1E12 IS THE CONVERSION
1352                                  ! TO MICROVOLTS/MHz
1354            Last_non_zero=Temp_r
1356          ELSE
1358            Temp_r=Last_non_zero
1360          END IF
1362          Temp_file(I-1,1)=Delta_x
1364          Temp_file(I-1,2)=Temp_r
1366          Delta_x=Freq_scale+Delta_x
1368        NEXT I
1370        CALL Data_to_disk_r(Temp_file(*),Num_o_pnts-1,Num_o_pnts-1,Data_id$)
1372        Intr_prty=Local_prty
1374        REDIM Temp_file(Number,2)
1376      END IF
1378      !
1380      C$="n"
1382    ! INPUT "Save the FFT of the response the waveform ? y/n",C$
1384      IF (C$="y") OR (C$="Y") THEN
1386        INPUT "Enter a 40 character or less data description.",Data_id$
1388        Intr_prty=Local_prty+3
1390        Delta_x=Freq_scale
1392        MAT Temp_comp= (1./Number)*Resp
1394        CALL Con_to_real(Temp_comp(*),Temp_file(*),0)
1396        IF Extended THEN
1398          REDIM Temp_file(Num_o_pnts,2)
1400        END IF
1402        CALL Data_to_disk_r(Temp_file(*),Num_o_pnts,Num_o_pnts,Data_id$)
1404        Intr_prty=Local_prty
1406        REDIM Temp_file(Number,2)
1408      END IF
1410      !
1412      C$="n"
1414      INPUT "Save the real part of the deconvolution result ? y/n",C$
1416      IF (C$="y") OR (C$="Y") THEN
1418        INPUT "Enter a 40 character or less data description.",Data_id$
1420        Intr_prty=Local_prty+3
1422        Delta_x=Time_scale
1424        CALL Con_to_real(Td(*),Temp_file(*),0)
1426        IF Extended THEN
1428          REDIM Temp_file(Num_o_pnts,2)
1430        END IF
1432        CALL Data_to_disk_r(Temp_file(*),Num_o_pnts,Num_o_pnts,Data_id$)
1434        Intr_prty=Local_prty
1436        REDIM Temp_file(Number,2)
1438      END IF
1440      !
1442      C$="n"
1444      INPUT "Save the imaginary part of the deconvolution result ? y/n",C$
```

```
1446      IF (C$="y") OR (C$="Y") THEN
1448         INPUT "Enter a 40 character or less data description.",Data_id$
1450         Intr_prty=Local_prty+3
1452         CALL Con_to_real(Td(*),Temp_file(*),1)
1454         IF Extended THEN
1456            REDIM Temp_file(Num_o_pnts,2)
1458         END IF
1460         CALL Data_to_disk_r(Temp_file(*),Num_o_pnts,Num_o_pnts,Data_id$)
1462         Intr_prty=Local_prty
1464         REDIM Temp_file(Number,2)
1466      END IF
1468      !
1470      DEALLOCATE Temp_file(*)
1472      SUBEXIT
1474   SUBEND
1476   !
1478   !*****************************************************************
1480   !
1482   SUB Load_disk_data(Basket_file(*),INTEGER Basketsize,Data_id$,INTEGER Flg)
1484 Load_disk_data:  ! Original: 13 Nov 1984
1486                  ! Revision: 02 Dec 1987
1488      !This routine will enter data files from the disk
1490      OPTION BASE 1
1492      !
1494      COM /Sys/ Sys_id$
1496      COM /History/ Status$[1],Time_orgn$[8],Date_orgn$[11]
1498      COM /History/ Time_chng$[8],Date_chng$[11],Description$[160]
1500   !
1502      COM /Labels/ Labels$(30)[60],INTEGER Lbl_count,REAL Lbl_addr(30,6)
1504   !Lbl_addr: x, y, pen, size, LDIR, LORG
1506   !
1508      COM /Data_param/ INTEGER Datacount,Filesize,Curvecount,Roster(17,4)
1510      COM /Data_param/ REAL Sym_size,Symbol$(17)[2],Curve_id$(17)[40]
1512      COM /Data_param/ REAL Xmin_data,Xmax_data
1514      COM /Data_param/ REAL Ymin_data,Ymax_data
1516   !
1518   !Roster: Curve#, Start Addr in File(*), Datacount, and PEN
1520   !Symbol$(i)="" or "Y" => no symbol, connect pts
1522   !Symbol$(i)="*Y" =>  * symbol, connect pts
1524   !Symbol$(i)="*N" =>  * symbol, do not connect pts
1526   !
1528      COM /Background/ Graphtype$[12],Margins$(2)[10],Papersize$[1]
1530      COM /Background/ REAL Pen_speed,INTEGER Backgnd_pen,Auto_time
1532      COM /Background/ INTEGER Auto_file,REAL X_cross_y,Y_cross_x
1534      COM /Background/ Xgrid_tick$[4],INTEGER Xmajor,Xminor
1536      COM /Background/ Ygrid_tick$[4],INTEGER Ymajor,Yminor
1538      COM /Background/ REAL Xmin_graph,Xmax_graph,Ymin_graph,Ymax_graph
1540   !
1542      COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
1544      COM /Interrupts/ INTEGER Intr_prty
1546      COM /Enlarge_file/ INTEGER Overflow
1548      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
1550      COM /Data_stuff/ INTEGER Number,REAL Delta_x,REAL Strt_time
1552   !
1554      INTEGER R,Hold_size,Local_prty,Allocated,Fls_cnt
1556      DIM Ac$[5],Tempfile$[10],Mask$[10],Ftype$[5],Fls$(1)[14]
```

```
1558      REAL Dtime
1560      OFF KEY
1562      Local_prty=Intr_prty
1564      !
1566      !Select the disk drive where the data exists
1568      !
1570      IF Overflow<>0 THEN Overflow=0
1572      Hold_size=0
1574      Dtime=0.
1576      Allocated=0
1578 Selectdrive:     !
1580      IF Diskdrive$="NO DISK" THEN Diskdrive$=""
1582      IF LEN(Diskdrive$)>0 THEN GOTO Choosefilename
1584      GRAPHICS OFF
1586      OUTPUT 2 USING "#,K";"K"
1588      CALL Select_disk
1590      IF Diskdrive$="NO DISK" THEN GOTO Mistakelineset
1592 Choosefilename: !
1594      Tempfile$=Filename$
1596      IF LEN(Filename$)>0 THEN GOTO Bring_in_data
1598      Ac$="CAT"
1600      CALL Enterfilename(Ac$)
1602      IF LEN(Filename$)=0 OR POS(Filename$,"*")>1 THEN
1604         IF POS(Filename$,"*")>1 THEN ! set mask$
1606            Mask$=Filename$[1,POS(Filename$,"*")-1]
1608            Filename$=""
1610         ELSE
1612            Mask$=""! no preselection
1614         END IF
1616         Ftype$="BDAT "  ! examine BDAT files only
1618         Fls_cnt=1  ! select one file
1620         Intr_prty=Local_prty+1
1622         CALL File_menu(Mask$,Ftype$,Fls$(*),Fls_cnt,0,0)
1624         Intr_prty=Local_prty
1626         Filename$=Fls$(1)
1628         IF LEN(Filename$)=0 THEN  ! aborted
1630            Filename$=Tempfile$
1632            GOTO Mistakelineset
1634         END IF
1636      END IF
1638 Bring_in_data: !
1640      !
1642      !Find this file on the disk.
1644      !
1646      ON ERROR GOTO Cant_findfile
1648      ASSIGN @Datapath TO Filename$&Diskdrive$
1650      OFF ERROR
1652      Dtime=TIMEDATE
1654      DISP " LOADING disk file: ";Filename$;" ... ";
1656      ON ERROR GOTO Bad_file
1658      ENTER @Datapath;Status$
1660      OFF ERROR
1662      ON ERROR GOTO Cant_findfile
1664      SELECT Status$
1666      CASE "Y"    ! All graphics/data parameters exist.REN 100,2
1668         DISP " Complete graph. "
```

```
1670        ENTER @Datapath;Time_orgn$,Date_orgn$
1672        ENTER @Datapath;Time_chng$,Date_chng$
1674        ENTER @Datapath;Description$
1676        ENTER @Datapath;Labels$(*),Lbl_count,Lbl_addr(*)
1678        ENTER @Datapath;Curve_id$(*),Symbol$(*)
1680        ENTER @Datapath;Roster(*),Curvecount
1682        ENTER @Datapath;Graphtype$,Margins$(*)
1684        ENTER @Datapath;X_cross_y,Y_cross_x
1686        ENTER @Datapath;Xgrid_tick$,Xmajor,Xminor
1688        ENTER @Datapath;Ygrid_tick$,Ymajor,Yminor
1690        ENTER @Datapath;Xmin_graph,Xmax_graph
1692        ENTER @Datapath;Ymin_graph,Ymax_graph
1694      CASE "N"    ! Only data parameters exist.
1696        DISP " RAW data. "
1698      CASE ELSE
1700 Bad_file: DISP CHR$(12)
1702        DISP "Data file is not recognized, entry aborted.";
1704        DISP " ...continue."
1706        BEEP
1708        PAUSE
1710        OFF ERROR
1712        GOTO Mistakelineset
1714      END SELECT
1716      !
1718      ENTER @Datapath;Data_id$
1720      IF Flg THEN
1722        ENTER @Datapath;Delta_x
1724        ENTER @Datapath;Datacount
1726        Hold_size=Datacount
1728      ELSE
1730        ENTER @Datapath;Datacount
1732        ENTER @Datapath;Hold_size
1734      END IF
1736      IF NOT Allocated THEN
1738        IF Datacount>=1 AND Hold_size>=1 THEN
1740          ALLOCATE Holding_file(Hold_size,2)
1742        ELSE
1744          ALLOCATE Holding_file(1,2)
1746        END IF
1748        Allocated=1
1750      END IF
1752      ENTER @Datapath;Holding_file(*)
1754      ASSIGN @Datapath TO *
1756      OFF ERROR
1758      IF NOT Flg THEN
1760        Delta_x=Holding_file(2,1)-Holding_file(1,1)
1762        Strt_time=Holding_file(1,1)
1764      END IF
1766      IF Datacount=0 THEN Mistakeline
1768      !
1770      !Copy data from Holding_file(*) to Basket_file(*)
1772      !
1774      MAT Basket_file= (0.)
1776      IF Datacount>Basketsize THEN  !Receiving file too small.
1778        Allocated=0
1780        DEALLOCATE Holding_file(*)
```

```
1782          DISP " DATA FILE overflow, new data discarded. ";
1784          DISP " (continue) "
1786          BEEP
1788          PAUSE
1790          IF Status$="Y" THEN
1792             Curvecount=0
1794             MAT Roster= (0)
1796          END IF
1798          Overflow=Hold_size
1800          GOTO Mistakelineset
1802       END IF
1804 Copydatafile:      !
1806       FOR R=1 TO Datacount
1808          Basket_file(R,1)=Holding_file(R,1)
1810          Basket_file(R,2)=Holding_file(R,2)
1812       NEXT R
1814       Basketsize=Datacount
1816       Flg=0
1818       GOTO Mistakeline
1820       !
1822 Mistakelineset:Datacount=0
1824       Flg=1
1826 Mistakeline:OFF KEY
1828       IF Allocated THEN DEALLOCATE Holding_file(*)
1830       LOOP
1832       EXIT IF TIMEDATE-Dtime>1.8
1834       END LOOP
1836       DISP CHR$(12)
1838       OUTPUT 2 USING "#,K";"K"
1840       SUBEXIT
1842       !
1844       ! ////////////////////////////////////////////////////
1846       !
1848 Cant_findfile:      !Error in searching for the file.
1850       BEEP 500,.6
1852       SELECT ERRN
1854       CASE 56
1856          DISP "That file does not exist on this disk  ";
1858       CASE 72,73,76,82
1860          DISP Diskdrive$;" has failed or is not available ";
1862       CASE ELSE
1864          DISP ERRM$;
1866       END SELECT
1868       DISP " ....CONTINUE to try again."
1870       PAUSE
1872       Filename$=""
1874       Diskdrive$=""
1876       GOTO Selectdrive
1878       !
1880    SUBEND
1882    !
1884    ! ***************************************************************
1886    !
1888    SUB Select_disk
1890 Select_disk: ! Original: 13 Nov 1984
1892              ! Revision: 02 Dec 1987
```

```
1894     OPTION BASE 1
1896     COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
1898     COM /Interrupts/ INTEGER Intr_prty
1900     COM /Sys_msi/ Msi_id$
1902     COM /Sys/ Sys_id$
1904     INTEGER Local_prty,Dd,Pt,Choose(1)
1906     DIM Disc$(30)[60],Title$[40],Displ$[60]
1908     Local_prty=Intr_prty
1910     OFF KEY
1912     !
1914     ! Define the disk drives available for this system, reserve the
1916     ! first characters for the drive address and the characters after
1918     ! the - for a description of the drive.
1920     !
1922     ! Example:
1924     ! Disc$(1)=":,700,0,0      HP 9133H HARD disk, volume 0."
1926     !
1928     !
1930     Displ$=" SELECT DISK DRIVE  ... Abort will cancel. "
1932     Title$=" Available disk drives for this system. "
1934     Pt=1   ! allow only one select
1936     !
1938     IF Diskdrive$[1,1]<>":" THEN Diskdrive$=""
1940     IF Msi_id$[1,1]<>":" THEN Msi_id$=SYSTEM$("MSI")
1942     IF Msi_id$[1,1]<>":" THEN ! Must be HFS subdirectory
1944        Ms_path$=Msi_id$[1,POS(Msi_id$,":")-1]  ! strip off subdirs
1946        IF Ms_path$[LEN(Ms_path$);1]<>"/" THEN Ms_path$=Ms_path$&"/"
1948        Msi_id$=Msi_id$[POS(Msi_id$,":"),LEN(Msi_id$)]
1950     END IF
1952     Diskdrive$=TRIM$(Diskdrive$)
1954     Msi_id$=TRIM$(Msi_id$)
1956     IF LEN(Diskdrive$)>0 AND LEN(Msi_id$)>0 THEN
1958        Disc$(1)=Diskdrive$&RPT$(" ",17-LEN(Diskdrive$))
1960        Disc$(1)=Disc$(1)&"- Last selected disk drive."
1962        Dd=1
1964        IF Diskdrive$<>Msi_id$ THEN
1966           Disc$(2)=Msi_id$&RPT$(" ",17-LEN(Msi_id$))
1968           Disc$(2)=Disc$(2)&"- Start-up mass storage unit specifier."
1970           Dd=Dd+1
1972        ELSE
1974           Disc$(1)=Disc$(1)&"  Start-up MSUS."
1976        END IF
1978     ELSE
1980        IF LEN(Msi_id$)>0 THEN
1982           Disc$(1)=Msi_id$&RPT$(" ",17-LEN(Msi_id$))
1984           Disc$(1)=Disc$(1)&"- Start-up mass storage unit specifier."
1986           Dd=1
1988        ELSE
1990           Dd=0
1992        END IF
1994     END IF
1996 Disk: !
1998     ! ..............  customize system drives here ..................
2000     ! Follow format with - after unit specifier, description is
2002     ! optional but recommended.
2004     ! ..............................................................
```

```
2006      !
2008      Disc$(Dd+1)=":,702,0          - HP 9122 dual microfloppy left drive"
2010      Disc$(Dd+2)=":,702,1          - HP 9122 dual microfloppy right drive"
2012      Disc$(Dd+3)=":,703,0          - HP 9125 single 5.25 floppy drive"
2014      Disc$(Dd+4)=":,1400           - HP 9133H hard disk volume 1"
2016      !
2018      Dd=Dd+4    ! add the number of drive specifiers above
2020      !
2022      IF Sys_id$[1,4]<>"S300" THEN
2024         Disc$(Dd+1)=":,4,1           - LEFT internal series 200"
2026         Disc$(Dd+2)=":,4,0           - RIGHT internal series 200"
2028         Dd=Dd+2
2030      END IF
2032      ! ..............................................................
2034      !
2036      CALL Menu_scroll(Displ$,Title$,Disc$(*),Dd,Pt,Choose(*))
2038      IF Pt=0 THEN
2040         Diskdrive$="NO DISK"
2042      ELSE
2044         Dd=POS(Disc$(Choose(Pt)),"-")-1  ! find -
2046         IF Dd>5 THEN  !  valid msus
2048            Diskdrive$=TRIM$(Disc$(Choose(Pt))[1,Dd])
2050         ELSE
2052            DISP " ERROR in reading MSUS from string, - chr not found. "
2054            BEEP
2056            CALL Pause_key_on
2058            Diskdrive$="NO DISK"
2060         END IF
2062      END IF
2064 Diskselected:OFF KEY
2066      SUBEXIT
2068   SUBEND
2070   !
2072   ! ********************************************************
2074   !
2076   SUB Enterfilename(Ac$)
2078 Enterfilename:  ! Original: 13 Nov 1984
2080              ! Revision: 10 Dec 1990  includes HFS directories
2082      OPTION BASE 1
2084      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
2086      COM /Interrupts/ INTEGER Intr_prty
2088      INTEGER I,Ascii_num,Maskflag,Namelength
2090      DIM Test$[256],Hfs_temp$[161]
2092      Namelength=10
2094      IF LEN(Ms_path$)>0 THEN OUTPUT KBD USING "K,#";"#"&Ms_path$&"H"
2096      DISP " ENTER HFS directory PATH (no file)";
2098      IF Ac$<>"PATH" THEN
2100         DISP ", ENTER / for HFS ROOT or null for LIF...";
2102      END IF
2104      LINPUT Hfs_temp$
2106      Hfs_temp$=TRIM$(Hfs_temp$)
2108      IF LEN(Hfs_temp$)>0 THEN
2110         IF LEN(Hfs_temp$)>1 AND Hfs_temp$[LEN(Hfs_temp$);1]<>"/" THEN
2112            Hfs_temp$=Hfs_temp$&"/"
2114         END IF
2116         IF LEN(Hfs_temp$)=1 THEN Hfs_temp$=""
```

```
2118        Namelength=14
2120      END IF
2122      IF Ac$="PATH" THEN
2124        Ms_path$=Hfs_temp$
2126        SUBEXIT
2128      END IF
2130      IF LEN(Filename$)>0 THEN OUTPUT KBD USING "K,#";"#"&Filename$&"H"
2132 Efn: !
2134      DISP " ENTER the FILE NAME ... ";
2136      SELECT Ac$
2138      CASE "CAT"
2140        DISP "(ENTER CAT mask* or ENTER null to CAT)";
2142      CASE "ABORT"
2144        DISP "(ENTER null to ABORT) ";
2146      CASE "VALID"
2148        DISP "(must be a VALID name!) ";
2150      END SELECT
2152      LINPUT Test$
2154      Test$=TRIM$(Test$)
2156      IF LEN(Test$)=0 AND Ac$="VALID" THEN GOTO Enterfilename
2158      IF LEN(Test$)=0 THEN Abortline
2160      IF LEN(Test$)>Namelength THEN
2162        BEEP
2164        DISP "ERROR in NAME ENTRY - max ";Namelength;" chars, you have ";
2166        DISP LEN(Test$);" "
2168        WAIT 1.8
2170        OUTPUT 2 USING "K,#";"#"&Test$&"H"
2172        GOTO Efn
2174      END IF
2176      IF POS(Test$,"*")>1 THEN
2178        Test$=Test$[1,POS(Test$,"*")-1]
2180        Maskflag=1
2182      ELSE
2184        Maskflag=0
2186      END IF
2188      FOR I=1 TO LEN(Test$)
2190        Ascii_num=NUM(Test$[I])
2192        SELECT Ascii_num
2194        CASE 65 TO 90,95,97 TO 122,48 TO 57
2196          !Allowed characters
2198        CASE ELSE
2200          BEEP
2202          DISP "ERROR in NAME ENTRY-ILLEGAL CHARACTERS, TRY AGAIN."
2204          WAIT 1.8
2206          OUTPUT 2 USING "K,#";"#"&Test$&"H"
2208          GOTO Efn
2210        END SELECT
2212      NEXT I
2214      IF Maskflag THEN
2216        Filename$=Test$&"*"
2218      ELSE
2220        Filename$=Test$
2222      END IF
2224      Ms_path$=Hfs_temp$
2226      SUBEXIT
2228 Abortline:Filename$=""
```

```
2230      IF Ac$="CAT" THEN Ms_path$=Hfs_temp$
2232      SUBEXIT
2234    SUBEND
2236    !
2238    ! ************************************************************
2240    !
2242    SUB File_menu(Mask$,Ftype$,Fls$(*),INTEGER Fls_cnt,Dir_on,Prt_on)
2244  File_menu:  !
2246      ! Original: 29 Jun 1987, G. Koepke
2248      ! Revision: 02 Dec 1987, 07:00
2250      OPTION BASE 1
2252      DEG
2254      COM /Sys/ Sys_id$[10]
2256      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
2258      COM /Interrupts/ INTEGER Intr_prty
2260      COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
2262      DIM Directory$(600)[80],Bd$(600)[71]
2264      DIM D$[80],T$[51],Ids$[40],Stat$[1],Test$[256]
2266      INTEGER Bd_cnt,File_cnt,I,C_cnt,C0(1),Format_error,End_search
2268      IF Fls_cnt>0 THEN ALLOCATE INTEGER Choose(Fls_cnt)
2270      !
2272      ! Catalog the disk specified
2274      !
2276      End_search=0
2278      REPEAT   ! Generate path to file and extract file name.
2280        ON ERROR GOTO Cat_errors
2282        DISP " Reading the Directory ... "
2284        IF LEN(Ms_path$)>0 THEN
2286          MASS STORAGE IS Ms_path$[1,LEN(Ms_path$)-1]&Diskdrive$
2288        ELSE
2290          MASS STORAGE IS Diskdrive$
2292        END IF
2294        CAT TO Directory$(*);NO HEADER,COUNT File_cnt
2296        OFF ERROR
2298        !
2300        ! set up array of legal file names.
2302        !
2304        Bd_cnt=0
2306        MAT Bd$= ("")
2308        FOR I=1 TO File_cnt
2310          SELECT Directory$(I)[32,36]
2312          CASE Ftype$          ! Ftype$ = "BDAT " or
2314                               ! Ftype$ = "PROG "
2316            IF LEN(Mask$)>0 THEN ! Test for mask$
2318              IF Directory$(I)[1,LEN(Mask$)]=Mask$ THEN
2320                Bd_cnt=Bd_cnt+1
2322                Bd$(Bd_cnt)=Directory$(I)[1;14]&" - "&Ftype$
2324              END IF
2326            ELSE
2328              Bd_cnt=Bd_cnt+1
2330              Bd$(Bd_cnt)=Directory$(I)[1;14]&" - "&Ftype$
2332            END IF
2334          CASE "DIR "          ! plus all "DIR " listings
2336            Bd_cnt=Bd_cnt+1
2338            Bd$(Bd_cnt)=Directory$(I)[1;14]&" - DIR "
2340          CASE ELSE
```

```
2342        END SELECT
2344      NEXT I
2346      IF LEN(Ms_path$)>0 AND Bd_cnt>0 AND Fls_cnt>0 THEN
2348        Bd_cnt=Bd_cnt+1
2350        Bd$(Bd_cnt)="----  MOVE back up ONE Directory level."
2352        Bd_cnt=Bd_cnt+1
2354        Bd$(Bd_cnt)="----  RETURN to ROOT Directory."
2356      END IF
2358      !
2360      ! set up file menu
2362      !
2364      D$=" Select "&VAL$(Fls_cnt)&" file name(s) for data entry."
2366      T$="List of "&Ftype$&"files and DIRs on "&Diskdrive$
2368      IF LEN(Mask$)>0 THEN
2370        T$=T$&" mask="&Mask$
2372      END IF
2374      IF Bd_cnt>0 THEN
2376        IF Dir_on>0 THEN GOSUB Read_data_id
2378        IF Prt_on THEN
2380          GOSUB List_directory
2382          End_search=1
2384        ELSE
2386          C_cnt=Fls_cnt
2388          DISP CHR$(12)
2390          IF Fls_cnt>0 THEN
2392            CALL Menu_scroll(D$,T$,Bd$(*),Bd_cnt,C_cnt,Choose(*))
2394          ELSE
2396            CALL Menu_scroll(D$,T$,Bd$(*),Bd_cnt,C_cnt,C0(*))
2398          END IF
2400          !
2402          ! transfer file names to Fls$(*).
2404          !
2406          IF C_cnt=0 THEN ! selection process aborted
2408            End_search=1
2410            MAT Fls$= ("")
2412          ELSE
2414            MAT SORT Choose(*)
2416            FOR I=1 TO C_cnt
2418              IF Bd$(Choose(I))[18,22]=Ftype$ THEN
2420                Fls$(I)=Bd$(Choose(I))[1;14]
2422                End_search=1
2424              ELSE  ! it must be a Directory or message.
2426                SELECT Bd$(Choose(I))[18,22]
2428                CASE "up ON" ! move up one directory
2430                  LOOP
2432                    Ms_path$=Ms_path$[1,LEN(Ms_path$)-1]
2434                  EXIT IF LEN(Ms_path$)=0
2436                    Test$=Ms_path$[LEN(Ms_path$);1]
2438                  EXIT IF Test$="/"
2440                  END LOOP
2442                CASE "ROOT "  ! jump to root directory
2444                  Ms_path$=""
2446                CASE "DIR  "  ! add directory to Ms_path$
2448                  Test$=TRIM$(Bd$(Choose(I))[1,14])
2450                  Ms_path$=Ms_path$&Test$&"/"
2452                CASE ELSE
```

```
                        DISP "ERROR in directory jump"
                        PAUSE
                     END SELECT
                     I=C_cnt
                  END IF
               NEXT I
            END IF
         END IF
      ELSE
         DISP " This directory contains no ";Ftype$;" files ... "
         WAIT 2.5
         End_search=1
      END IF
      DISP CHR$(12)
   UNTIL End_search
   SUBEXIT
Cat_errors:!
   DISP " ERROR ... ";ERRM$
   BEEP
   CALL Pause_key_on
   DISP CHR$(12)
   C_cnt=0
   MAT Fls$= ("")
   SUBEXIT
   !
   ! ////////////////////////////////////////////////
   !
Read_data_id: ! This routine expects to see Ids$ from
            ! GRAPH_DATA raw data files.
   DISP " Reading file contents ... Please stand by. "
   PRINT TABXY(1,18);" Reading #";
   FOR I=1 TO Bd_cnt    ! each BDAT file
      PRINT TABXY(11,18);
      PRINT USING "3D,4A,3D,2A,#";I," of ",Bd_cnt,". "
      Ids$="Data not recognized."
      IF Bd$(I)[18,22]="BDAT " THEN
         ON ERROR GOTO Not_recognized
         ASSIGN @Io_path TO Bd$(I)[1;14]
         ENTER @Io_path;Stat$
         SELECT Stat$
         CASE "N"
            ENTER @Io_path;Ids$
         CASE "Y"
            Ids$="Complete graph in GRAPH_DATA form."
         END SELECT
Not_recognized:ASSIGN @Io_path TO *
         OFF ERROR
         IF Dir_on=2 THEN
            GOSUB Interpret_1
            IF Format_error THEN GOTO Other_format
            GOTO Go_on
         END IF
Other_format:!
         Bd$(I)[23,71]=" ... "&Ids$
      END IF
Go_on:NEXT I
```

```
2566      PRINT TABXY(1,18);RPT$(" ",40);
2568      DISP CHR$(12);
2570      RETURN
2572      !
2574      ! ////////////////////////////////////////////////
2576      !
2578 Interpret_1: ! This is used to interpret ID strings.
2580      Format_error=1
2582      ! identify this particular format
2584      RETURN
2586      !
2588      ! ////////////////////////////////////////////////
2590      !
2592 List_directory: ! This routine will provide a tabular listing of
2594               ! the directory along with lds$ if provided
2596               !
2598      DISP " Listing directory ... "
2600      ON TIMEOUT 7,10 GOTO Printer_kaput
2602      PRINTER IS Printer
2604      PRINT USING "//"
2606      PRINT T$
2608      IF LEN(Ms_path$)>0 THEN PRINT "HFS Path:  ";Ms_path$
2610      PRINT RPT$("=",80)
2612      PRINT "File name";
2614      IF Dir_on THEN
2616         PRINT "      - TYPE  ... contents"
2618      ELSE
2620         PRINT "      - TYPE"
2622      END IF
2624      PRINT RPT$("-",80)
2626      FOR I=1 TO Bd_cnt
2628         IF Bd$(I)[18,22]=Ftype$ OR Bd$(I)[18,22]="DIR  " THEN
2630            PRINT Bd$(I)
2632         END IF
2634      NEXT I
2636      PRINT RPT$("_",80)
2638      PRINT
2640      PRINTER IS CRT
2642      OFF TIMEOUT 7
2644      RETURN
2646 Printer_kaput:DISP " Printer not responding ... listing aborted. "
2648      BEEP
2650      WAIT 1.8
2652      OFF TIMEOUT 7
2654      RETURN
2656   SUBEND
2658   !
2660   ! ************************************************************
2662   !
2664   SUB Menu_scroll(D$,T$,Items$(*),INTEGER Item_cnt,To_select,Choose(*))
2666 Menu_scroll:! Original: 22 Jun 1987, Galen Koepke, NBS 723.04
2668            ! Revision: 22 Aug 1990, 12:00, Dennis Camell
2670            !
2672            ! A general purpose menu utility for scrolling items and
2674            ! selecting either a fixed number or a random number
2676            ! of items.
```

```
2678        ! for fixed  : To_select > 0
2680        ! for random : To_select = -1
2682        ! The items are arranged in screens of 15 items each and
2684        ! the user may access screens via softkeys. There may be
2686        ! up to 40 screens or 600 items to choose from.
2688        ! Maximum sizes: D$[80], T$[51], Items(*)[70]
2690        ! Items$(*) contains the item descriptions
2692        ! Item_cnt is the number of items in Items$(*)
2694        ! Choose(*) is dimensioned to the number of required choices
2696        !         and will be filled with the item numbers chosen.
2698        ! To_select is the number of required choices.
2700        !
2702      OPTION BASE 1
2704      PRINTER IS CRT
2706      DEG
2708      GOSUB Def_variables
2710      GOSUB Define_screens
2712      GOSUB Make_selections
2714      IF Null_file THEN  ! reset to zero
2716         Item_cnt=0
2718         Items$(1)=""
2720         To_select=0    ! no valid selections
2722      END IF
2724      SUBEXIT
2726      !
2728      ! ////////////////////////////////////////////////
2730      !
2732 Def_variables:!
2734      COM /Interrupts/ INTEGER Intr_prty
2736      COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
2738      COM /Sys/ Sys_id$[10]
2740      !
2742      INTEGER Screen_cnt,Items_per_scn,First_item(40),Last_item(40)
2744      INTEGER I,J,K,First_line,Last_line,Active_screen,Pointer,Last_pt
2746      INTEGER Local_prty,Skips,Knobcount,Pointeractive,K0,Null_file
2748      INTEGER Exit_flag,Temp,Random_select,Indx
2750      DIM Marker$[8],Test$[256]
2752      !
2754      ! initialize parameters
2756      !
2758      Local_prty=Intr_prty
2760      IF Local_prty<1 THEN Local_prty=10
2762      IF LEN(Sys_id$)=0 THEN Sys_id$=SYSTEM$("SYSTEM ID")
2764      IF Item_cnt<1 THEN
2766         Null_file=1
2768         Item_cnt=1
2770         To_select=0
2772         Items$(1)="**** Empty ****"
2774      ELSE
2776         Null_file=0
2778      END IF
2780      IF To_select=-1 THEN
2782         Random_select=1   ! choose random number of items
2784         To_select=0       ! needed for softkeys
2786      END IF
2788      IF To_select>Item_cnt THEN To_select=Item_cnt
```

```
2790      MAT Choose= (999)
2792      Skips=0
2794      Knobcount=0
2796      Doneflag=0
2798      Marker$="====>"&RPT$(CHR$(8),4)
2800      RETURN
2802      !
2804      ! ////////////////////////////////////////////////
2806      !
2808 Define_screens:! Set up screens of 15 items each.
2810            !
2812      Items_per_scn=15    ! Maximum number of displayable items
2814      IF INT(Item_cnt/Items_per_scn)=Item_cnt/Items_per_scn THEN
2816         Screen_cnt=INT(Item_cnt/Items_per_scn)
2818      ELSE
2820         Screen_cnt=INT(Item_cnt/Items_per_scn)+1
2822      END IF
2824      J=1
2826      FOR I=1 TO Screen_cnt   ! set up each screen
2828         First_item(I)=J
2830         IF J+Items_per_scn-1<Item_cnt THEN
2832            Last_item(I)=J+Items_per_scn-1
2834            J=J+Items_per_scn
2836         ELSE
2838            Last_item(I)=Item_cnt
2840         END IF
2842      NEXT I
2844      RETURN
2846      !
2848      ! ////////////////////////////////////////////////
2850      !
2852 Make_selections:! MENU setup and use.
2854      Active_screen=1   ! first screen is active
2856      First_line=2      ! first printed line on screen = 2 or greater.
2858      GOSUB Write_screen  ! activate screen at Active_screen
2860                          ! and set First_line and Last_line for Pointer
2862                          ! write Marker$ to first non-selected line.
2864      K0=0                ! Keys start at zero
2866      Exit_flag=0         ! allow ENTER key to exit when selections filled.
2868 Key_loop:  !
2870      ON KBD,Local_prty GOSUB Process_kbd
2872      ON KNOB .01,Local_prty GOSUB Move_pointer
2874      IF Random_select THEN
2876         ! set keys for random selection
2878         DISP D$
2880         ON KEY K0 LABEL " Select",Local_prty GOSUB Select_random
2882         ON KEY K0+9 LABEL " Accept",Local_prty GOTO Exit_line
2884      ELSE ! set key K0 for fixed selection
2886         IF Skips<To_select THEN
2888            DISP D$
2890            IF To_select>1 THEN
2892               Test$=" Select "&VAL$(Skips+1)&" of "&VAL$(To_select)
2894            ELSE
2896               Test$=" Select"
2898            END IF
2900            ON KEY K0 LABEL Test$,Local_prty GOSUB Select_fixed
```

```
2902        ELSE
2904          IF To_select>0 THEN
2906            DISP " Selection process complete ..."
2908          ELSE
2910            DISP " Menu for information only ... "
2912          END IF
2914          ON KEY K0 LABEL "Accept",Local_prty GOTO Exit_line
2916        END IF
2918      END IF
2920      IF Active_screen<Screen_cnt THEN
2922        ON KEY K0+1 LABEL " Next  Screen",Local_prty GOSUB Next_screen
2924      ELSE
2926        OFF KEY K0+1
2928      END IF
2930      IF Active_screen>1 THEN
2932        ON KEY K0+2 LABEL " Last  Screen",Local_prty GOSUB Last_screen
2934      ELSE
2936        OFF KEY K0+2
2938      END IF
2940      IF Skips>0 OR Random_select THEN
2942        ON KEY K0+3 LABEL " Reset  Select",Local_prty GOSUB Select_reset
2944      ELSE
2946        OFF KEY K0+3
2948      END IF
2950      IF To_select>0 OR Random_select THEN
2952        ON KEY K0+4 LABEL " Abort ",Local_prty GOTO Escape_line
2954      ELSE
2956        OFF KEY K0+4
2958      END IF
2960      IF Screen_cnt>2 THEN
2962        ON KEY K0+6 LABEL "Jump to Screen",Local_prty GOSUB Jump_to_scn
2964      ELSE
2966        OFF KEY K0+6
2968      END IF
2970      IF Exit_flag THEN Exit_line
2972      GOTO Key_loop
2974 Escape_line:Skips=0
2976      MAT Choose= (0)
2978      To_select=0
2980 Exit_line:OFF KEY
2982      MAT SORT Choose(*)
2984      OFF KNOB
2986      OFF KBD
2988      OUTPUT KBD;CHR$(255)&CHR$(75);
2990      PRINT CHR$(128);
2992      ! everything cleared, now go back to work.
2994      RETURN
2996      !
2998      ! ////////////////////////////////////////////////
3000      !
3002 Next_screen:     !
3004      OFF KBD
3006      OFF KNOB
3008      OFF KEY
3010      IF Active_screen=Screen_cnt THEN RETURN
3012      Active_screen=Active_screen+1
```

```
3014      GOSUB Write_screen
3016      RETURN
3018      !
3020      ! ////////////////////////////////////////////////
3022      !
3024 Last_screen:      !
3026      OFF KBD
3028      OFF KNOB
3030      OFF KEY
3032      IF Active_screen=1 THEN RETURN
3034      Active_screen=Active_screen-1
3036      GOSUB Write_screen
3038      RETURN
3040      !
3042      ! ////////////////////////////////////////////////
3044      !
3046 Jump_to_errors:DISP " Not a valid screen number ... try again. "
3048      BEEP
3050      WAIT 1.8
3052 Jump_to_scn: !
3054      OFF KBD
3056      OFF KNOB
3058      OFF KEY
3060      DISP " ENTER the screen number desired (1 to ";Screen_cnt;").";
3062      LINPUT Test$
3064      Test$=TRIM$(Test$)
3066      IF LEN(Test$)=0 THEN Jump_to_return
3068      ON ERROR GOTO Jump_to_errors
3070      Temp=INT(VAL(Test$))
3072      OFF ERROR
3074      IF Temp<1 OR Temp>Screen_cnt THEN Jump_to_errors
3076      Active_screen=Temp
3078      GOSUB Write_screen
3080 Jump_to_return: !
3082      DISP CHR$(12)
3084      Test$=""
3086      RETURN
3088      !
3090      ! ////////////////////////////////////////////////
3092      !
3094 Select_fixed:!
3096      OFF KBD
3098      OFF KNOB
3100      OFF KEY
3102      IF NOT Pointeractive THEN
3104         DISP "NO additional selections for this screen."
3106         BEEP
3108         WAIT 2
3110         DISP CHR$(12);
3112         RETURN
3114      END IF
3116      IF Skips=To_select THEN
3118         IF To_select=0 THEN
3120            DISP "This menu is for information only,";
3122            DISP " no selection allowed."
3124         ELSE
```

```
3126            DISP "All selections have been filled,";
3128            DISP " 'Select Reset' to repeat."
3130          END IF
3132          BEEP
3134          WAIT 2
3136          DISP CHR$(12);
3138          RETURN
3140        END IF
3142        Skips=Skips+1
3144        Choose(Skips)=First_item(Active_screen)+Pointer-First_line
3146        PRINT CHR$(129);  ! inverse video
3148        PRINT TABXY(10,Pointer);Items$(Choose(Skips))
3150        PRINT CHR$(128);
3152        PRINT TABXY(1,Pointer);
3154        SELECT Pointer
3156        CASE First_line
3158          GOSUB Point_forward
3160        CASE Last_line
3162          GOSUB Point_backward
3164        CASE ELSE
3166          ! move forward unless it requires wrapping to beginning.
3168          IF Skips-1>0 THEN  ! check for selected items.
3170            I=Pointer-First_line
3172            LOOP
3174              K=0
3176              FOR J=1 TO Skips
3178                IF First_item(Active_screen)+I=Choose(J) THEN K=1
3180              NEXT J
3182            EXIT IF K=0
3184              I=I+1
3186              IF I+First_line>Last_line THEN K=-1
3188            EXIT IF K=-1
3190            END LOOP
3192            IF K=0 THEN
3194              GOSUB Point_forward
3196            ELSE
3198              GOSUB Point_backward
3200            END IF
3202          ELSE
3204            GOSUB Point_forward
3206          END IF
3208        END SELECT
3210        RETURN
3212        !
3214        ! ////////////////////////////////////////////////
3216        !
3218 Select_random:!
3220        OFF KBD
3222        OFF KNOB
3224        OFF KEY
3226        Test$="NO"
3228        IF NOT Pointeractive THEN
3230          DISP "NO additional selections for this screen."
3232          BEEP
3234          WAIT 2
3236          DISP CHR$(12);
```

```
3238        RETURN
3240      END IF
3242      FOR I=1 TO To_select
3244        IF Choose(I)=First_item(Active_screen)+Pointer-First_line  THEN
3246          Indx=I
3248          Test$="YES"
3250        END IF
3252      NEXT I
3254      SELECT Test$
3256      CASE "YES"              ! Selected item is tagged ... untag
3258        IF Pointer<>Last_item(Active_screen)+1  AND Pointer<>17 THEN
3260          PRINT CHR$(128);! normal video
3262        ELSE
3264          PRINT CHR$(132);! underline video
3266        END IF
3268        PRINT TABXY(10,Pointer);Items$(Choose(Indx))
3270        FOR I=Indx TO To_select-1
3272          Choose(I)=Choose(I+1)
3274        NEXT I
3276        Choose(To_select)=999
3278        To_select=To_select-1
3280      CASE "NO"               ! Selected item is untagged ... tag it
3282        To_select=To_select+1
3284        Choose(To_select)=First_item(Active_screen)+Pointer-First_line
3286        IF Pointer<>Last_item(Active_screen)+1  AND Pointer<>17 THEN
3288          PRINT CHR$(129);! inverse video
3290        ELSE
3292          PRINT CHR$(133);! inverse video with underline
3294        END IF
3296        PRINT TABXY(10,Pointer);Items$(Choose(To_select))
3298      END SELECT
3300      PRINT CHR$(128);
3302      PRINT TABXY(1,Pointer);
3304      RETURN
3306      !
3308      ! ////////////////////////////////////////////////
3310      !
3312 Select_reset:      !Clear Choose file
3314      OFF KBD
3316      OFF KNOB
3318      OFF KEY
3320      IF Random_select THEN To_select=0
3322      Skips=0
3324      MAT Choose= (999)
3326      GOSUB Write_screen
3328      RETURN
3330      !
3332      ! ////////////////////////////////////////////////
3334      !
3336 Process_kbd:! Allow use of arrows and enter key in addition to soft.
3338      Test$=KBD$
3340      IF LEN(Test$)=1 AND Test$[1,1]<>CHR$(32) THEN
3342        BEEP 80.,.1
3344        RETURN
3346      END IF
3348      IF Test$[1,1]=CHR$(32) THEN GOSUB Point_forward
```

```
3350      IF Test$[1,1]<>CHR$(255) THEN RETURN
3352      SELECT Test$[2,2]
3354      CASE CHR$(255)
3356            ! do nothing
3358      CASE "V","T"
3360         GOSUB Point_forward
3362      CASE "^","W"
3364         GOSUB Point_backward
3366      CASE "E","s","t","&"
3368         IF Random_select THEN
3370            GOSUB Select_random
3372         ELSE
3374            IF Skips<To_select THEN
3376               GOSUB Select_fixed
3378            ELSE
3380            ! exit routine
3382               Exit_flag=1
3384            END IF
3386         END IF
3388      CASE ELSE
3390         BEEP 80.,.1
3392      END SELECT
3394      Test$=""
3396      RETURN
3398      !
3400      ! ////////////////////////////////////////////////
3402      !
3404 Point_forward:Knobcount=5
3406      GOSUB Move_pointer
3408      RETURN
3410 Point_backward:Knobcount=-5
3412      GOSUB Move_pointer
3414      RETURN
3416      !
3418      ! ////////////////////////////////////////////////
3420      !
3422 Jog_pointer:! Move the selection pointer on the active screen.
3424            ! without regard to selected values
3426      IF Knobcount>0 THEN  ! Move forward
3428         Pointer=Pointer+1
3430      ELSE                 ! Move backward
3432         Pointer=Pointer-1
3434      END IF
3436      IF Pointer<First_line THEN Pointer=Last_line
3438      IF Pointer>Last_line THEN Pointer=First_line
3440      RETURN
3442      !
3444      ! ////////////////////////////////////////////////
3446      !
3448 Move_pointer:! Control pointer to avoid re-selection of items
3450      IF NOT Pointeractive THEN RETURN  ! No selections to be made.
3452      Knobcount=Knobcount+KNOBX-KNOBY
3454      IF ABS(Knobcount)<4 THEN RETURN
3456      Last_pt=Pointer
3458      GOSUB Jog_pointer
3460      IF Skips>0 THEN
```

```
3462        LOOP
3464           J=Pointer-First_line
3466           FOR I=1 TO Skips
3468              IF First_item(Active_screen)+J=Choose(I) THEN J=999
3470           NEXT I
3472           IF J=999 AND Pointer=Last_pt THEN Pointeractive=0
3474        EXIT IF Pointeractive=0
3476           IF J=999 THEN GOSUB Jog_pointer
3478        EXIT IF J<>999
3480        END LOOP
3482     END IF
3484     Knobcount=0
3486     OUTPUT KBD;CHR$(255)&CHR$(84);  ! Bring screen home
3488     IF Last_pt=Last_line THEN PRINT CHR$(132);
3490     PRINT "    ";
3492     IF Pointeractive THEN ! Pointer active
3494        IF Pointer=Last_line THEN
3496           PRINT CHR$(132);
3498        ELSE
3500           PRINT CHR$(128);
3502        END IF
3504        PRINT TABXY(1,Pointer);Marker$;CHR$(128);
3506     END IF
3508     RETURN
3510     !
3512     ! ////////////////////////////////////////////////
3514     !
3516 Write_screen:! Write the screen pointed to by Active_screen
3518     ! home and clear screen
3520     OUTPUT KBD;CHR$(255)&CHR$(84)&CHR$(255)&CHR$(75);
3522     Knobcount=KNOBX-KNOBY    ! Clear knob and keyboard
3524     Knobcount=0
3526     Test$=KBD$
3528     Test$=""
3530     !
3532     PRINT TABXY(1,First_line-1);CHR$(132);" Item #| Screen #";
3534     PRINT USING "#,2D,4A,2D,3A";Active_screen," of ";Screen_cnt;" | "
3536     PRINT T$;RPT$(" ",51-LEN(T$));
3538     PRINT TABXY(80,First_line-1);"|";CHR$(128);
3540     J=0
3542     REPEAT
3544        IF J=Last_item(Active_screen)-First_item(Active_screen) THEN
3546           PRINT CHR$(132);
3548           PRINT TABXY(1,First_line+J);RPT$(" ",80)
3550        ELSE
3552           PRINT CHR$(128);
3554        END IF
3556        PRINT TABXY(5,First_line+J);
3558        PRINT USING "3D,A,#";First_item(Active_screen)+J,"|"
3560        IF Random_select THEN
3562           FOR I=1 TO To_select
3564              IF First_item(Active_screen)+J=Choose(I) THEN
3566                 PRINT CHR$(129);
3568              END IF
3570           NEXT I
3572        ELSE
```

```
3574            IF Skips>0 THEN ! make this line inverse video
3576               FOR I=1 TO Skips
3578                  IF First_item(Active_screen)+J=Choose(I) THEN
3580                     PRINT CHR$(129);
3582                  END IF
3584               NEXT I
3586            END IF
3588         END IF
3590         PRINT TABXY(10,First_line+J);Items$(First_item(Active_screen)+J)
3592         PRINT TABXY(80,First_line+J);"|";
3594         J=J+1
3596      UNTIL J>=(Last_item(Active_screen)-First_item(Active_screen)+1)
3598      Last_line=Last_item(Active_screen)-First_item(Active_screen)
3600      Last_line=Last_line+First_line
3602      !
3604      ! set marker to first non-selected item.
3606      !
3608      Pointeractive=0
3610      IF To_select>0 OR Random_select THEN Pointeractive=1
3612      IF Skips>0 AND Pointeractive=1 THEN ! find first non-selected item
3614         J=0
3616         LOOP
3618            Pointer=First_line+J
3620            FOR I=1 TO Skips
3622               IF First_item(Active_screen)+J=Choose(I) THEN Pointer=0
3624            NEXT I
3626         EXIT IF Pointer<>0
3628            J=J+1
3630            IF First_line+J>Last_line THEN
3632               Pointeractive=0
3634               Pointer=First_line
3636            END IF
3638         EXIT IF Pointer<>0
3640         END LOOP
3642      ELSE
3644         Pointer=First_line
3646      END IF
3648      IF Pointeractive THEN
3650         IF Pointer=Last_line THEN
3652            PRINT CHR$(132);
3654         ELSE
3656            PRINT CHR$(128);
3658         END IF
3660         PRINT TABXY(1,Pointer);Marker$;CHR$(128);
3662      END IF
3664      RETURN
3666   SUBEND
3668   !
3670   ! ************************************************************
3672   !
3674   SUB Data_to_disk_r(REAL File(*),INTEGER Filesize,Datacount,Data_id$)
3676 Data_to_disk_r:  ! Original: 13 Nov 1984
3678                  ! Revision: 06 Aug 1987
3680      ! This routine will SAVE data files on the disk in RAW data format.
3682      ! Special features:
3684      !    If the Diskdrive$ and/or the Filename$ are null this routine
```

```
3686      !    will prompt the operator for information.  However, if they
3688      !    are not null it is assumed that the program is supplying the
3690      !    correct information.
3692      !
3694      OPTION BASE 1
3696      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
3698      COM /Interrupts/ INTEGER Intr_prty
3700      INTEGER Local_prty,Diskspace
3702      DIM Ac$[5],Status$[1]
3704      REAL Dtime
3706      OFF KEY
3708      Local_prty=Intr_prty
3710      Dtime=0.
3712      !
3714      !Select the disk drive for data storage
3716      !
3718 Selectdrive: !
3720      IF Diskdrive$="NO DISK" THEN Diskdrive$=""
3722      IF LEN(Diskdrive$)>0 THEN GOTO Choosefilename
3724      GRAPHICS OFF
3726      OUTPUT 2 USING "#,K";"K"
3728      CALL Select_disk
3730      IF Diskdrive$="NO DISK" THEN GOTO Mistakeline
3732 Choosefilename: !
3734      IF LEN(Filename$)>0 THEN GOTO Send_to_disk
3736      Ac$="ABORT"
3738      CALL Enterfilename(Ac$)
3740      IF LEN(Filename$)=0 THEN GOTO Mistakeline
3742 Send_to_disk: ! Create file and save information.
3744      ON ERROR GOTO Cant_savedata
3746      Diskspace=INT((Filesize*16.0)/256)+2
3748      CREATE BDAT Filename$&Diskdrive$,Diskspace,256
3750      Dtime=TIMEDATE
3752      DISP " SAVING data in file ";Filename$;" on ";Diskdrive$
3754      Status$="N"
3756      ASSIGN @Datapath TO Filename$&Diskdrive$
3758      OUTPUT @Datapath;Status$
3760      OUTPUT @Datapath;Data_id$    !40 chrs description of data
3762      OUTPUT @Datapath;Datacount   !number of xy points
3764      OUTPUT @Datapath;Filesize    !size of array
3766      OUTPUT @Datapath;File(*)
3768      ASSIGN @Datapath TO *
3770      OFF ERROR
3772      !
3774 Mistakeline:OFF KEY
3776      LOOP
3778      EXIT IF TIMEDATE-Dtime>1.8
3780      END LOOP
3782      DISP CHR$(12)
3784      OUTPUT 2 USING "#,K";"K"
3786      SUBEXIT
3788      !
3790      ! ////////////////////////////////////////////////////
3792      !
3794 Cant_savedata:  !
3796      BEEP 500,.6
```

```
3798     SELECT ERRN
3800     CASE 72,73,76,78,81,82,90,93
3802        DISP Diskdrive$;" has failed or is not available ";
3804        DISP " ....CONTINUE to try again."
3806        PAUSE
3808        Diskdrive$=""
3810     CASE 84,85
3812        DISP " This disk is not initialized ";
3814        DISP " ....CONTINUE to try again."
3816        PAUSE
3818        Diskdrive$=""
3820     CASE 55,64
3822        DISP " This disk is full, insert new floppy and/or";
3824        DISP " select new drive ...CONTINUE "
3826        PAUSE
3828        Diskdrive$=""
3830     CASE ELSE
3832        CALL Errortrap
3834        IF LEN(Filename$)>0 THEN GOTO Send_to_disk
3836     END SELECT
3838     GOTO Selectdrive
3840     !
3842  SUBEND
3844  !
3846  ! ***************************************************************
3848  !
3850  SUB Pause_key_on
3852 Pause_key_on: ! Make sure that CONTINUE key exists.
3854           ! Original: 02 Dec 1987
3856           ! Revision: 02 Dec 1987
3858     OPTION BASE 1
3860     COM /Sys/ Sys_id$[10]
3862     IF Sys_id$[1,4]="S300" THEN ! reset to S300 system keys
3864        CONTROL KBD,15;0
3866        CONTROL CRT,12;2
3868        LOAD KEY
3870     END IF
3872     PAUSE
3874     IF Sys_id$[1,4]="S300" THEN  ! set to S200 compatible keys
3876        OUTPUT KBD USING "K,#";"SCRATCH KEYX"
3878        CONTROL KBD,15;1
3880        CONTROL CRT,12;0
3882     END IF
3884     SUBEXIT
3886  SUBEND
3888  !
3890  ! ***************************************************************
3892  !
4020  SUB Errortrap
4022 Errortrap: ! Original: 13 Nov 1984
4024           ! Revision: 02 Dec 1987
4026     ! Trap most errors here
4028     OPTION BASE 1
4030     COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
4032     DIM File$[20],Test$[256],What$[20],Ac$[5]
4034     BEEP 400,.6
```

```
4036     SELECT ERRN
4038     CASE 54
4040        DISP "DUPLICATE FILE NAME:  ";Filename$;
4042        DISP "....PURGE old one? (Y/N)";
4044        LINPUT What$
4046        What$ = TRIM$(What$)
4048        SELECT What$[1,1]
4050        CASE "Y","y"
4052           PURGE Ms_path$&Filename$&Diskdrive$
4054        CASE ELSE
4056           Ac$ = "VALID"
4058           CALL Enterfilename(Ac$)
4060        END SELECT
4062     CASE 52,53
4064        DISP "Improper FILE NAME -- ENTER NEW FILE NAME";
4066        OUTPUT 2 USING "#,K,K";"#";Filename$
4068        LINPUT Filename$
4070        Filename$ = TRIM$(Filename$)
4072     CASE 56
4074        DISP "FILE:  ";Filename$;" is not on this disk, please insert";
4076        DISP " correct disk"
4078        CALL Pause_key_on
4080     CASE 64
4082        DISP "This disk is full, PLEASE insert clean disk"
4084        CALL Pause_key_on
4086     CASE 56
4088        DISP "DATA INPUT disk must be in drive!!  ";
4090        DISP "...CONTINUE when ready."
4092        CALL Pause_key_on
4094     CASE 72,73,76
4096        DISP Diskdrive$;
4098        DISP " is not available, type correct";
4100        DISP " unit specifier (ie. ':,707,0')."
4102        OUTPUT 2 USING "K,#";Diskdrive$
4104        LINPUT Diskdrive$
4106     CASE 80
4108        DISP "CHECK DISK drive door!"
4110        CALL Pause_key_on
4112     CASE ELSE
4114        DISP ERRM$;" 'CONTINUE' when fixed"
4116        CALL Pause_key_on
4118     END SELECT
4120     DISP CHR$(12)
4122     SUBEXIT
4124  SUBEND
4126  !
4128  ! ***************************************************************
4130  !
```

```
100  ! RE-STORE "FIXACQ:,1400"
102  !
104   COM /Sys/ Sys_id$[10]
106   COM /Sys_msi/ Msi_id$[20]
108  !
110   OUTPUT KBD USING "K,#";"SCRATCH KEYE"        !ERASE SOFT KEYS
112   CONTROL KBD,15;0! sets the color of the soft keys
114   CONTROL KBD,2;1
116  !
118  !*****************************************************************
120  ! Program by S.M. Chesnut. The National Institute of Standards
122  ! and Technology. Based on a program by W. Gans and R. Stafford.
124  !
126 Date_line:  !
128          !*************************************************
130          ! Last Modified May 17,1991 by S.M.C
132          !*************************************************
134  !
136  !*****************************************************************
138  !
140   Intr_prty=1
142   CALL Fixacq
144  !
146   MASS STORAGE IS ":,1400"! Resets the mass storage device to
148                    ! the hard drive. This number may be
150                    ! changed to suit.
152   OUTPUT KBD USING "K,#";"LOAD KEYE"! restore the typing aid keys
154   PRINT TABXY(1,5);"END of program. So long."
156  !
158   END
160  !
162  !_____________________________________________________________
164  !
166  !
168   SUB Fixacq
170   !
172 Fixacq:   !
174   !
176      OPTION BASE 1
178      RAD
180   ! This program reads in the following data:
182   ! The device under test (DUT) waveform, Wave,
184   ! the voltage calibration data ,Vcal,
186   ! and the time calibration data, Tcal.
188   ! The DUT data is then "fixed" using the calibration data.
190   !
192      COM /Interrupts/ INTEGER Intr_prty
194      COM /Sys_msi/ Msi_id$[20]
196      COM /Sys/ Sys_id$[10]
198      COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
200      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
202   !
204      Sys_id$=SYSTEM$("SYSTEM ID")
206      Msi_id$=SYSTEM$("MSI")
208   !
```

```
210       DIM Vcal(100,2),Tcal(100,2),Wave(4096,2)
212       INTEGER Datacount,Flg,Vfilesize,Tfilesize,Local_prty,Stp_flg
214       DIM Data_id$[40],Ch_vcal$[1],Ch_tcal$[1]
216       !
218       Datacount=32767
220       Vfilesize=4096
222       Tfilesize=4096
224       Flg=0
226       Ch_vcal$=""
228       Ch_tcal$=""
230       Local_prty=Intr_prty
232       Intr_prty=Intr_prty+2
234       Filename$=""
236       Diskdrive$=""
238 Dut:   DISP "Input DUT waveform file name when prompted"
240       WAIT 1.0
242       CALL Load_disk_data(Wave(*),Datacount,Data_id$,Flg)
244       IF NOT Flg THEN
246          BEEP
248          DISP "NO FILE IN MEMORY, TRY AGAIN."
250          WAIT 1.5
252          GOSUB Reset_filestuff
254          GOTO Dut
256       END IF
258       GOSUB Reset_filestuff
260       REDIM Wave(Datacount,2)
262       INPUT "Is voltage calibration desired? y/n",Ch_vcal$
264       IF Ch_vcal$="Y" OR Ch_vcal$="y" THEN
266 Vcaldata:  DISP "Input voltage calibration file name when prompted"
268          WAIT 1.0
270          CALL Load_disk_data(Vcal(*),Vfilesize,Data_id$,Flg)
272          IF NOT Flg THEN
274             BEEP
276             DISP "NO FILE IN MEMORY, TRY AGAIN."
278             WAIT 1.5
280             GOSUB Reset_filestuff
282             GOTO Vcaldata
284          END IF
286          GOSUB Reset_filestuff
288       END IF
290       INPUT "Is time scale calibration desired? y/n",Ch_tcal$
292       IF Ch_tcal$="Y" OR Ch_tcal$="y" THEN
294 Tcaldata:  DISP "Input time calibration file name when prompted"
296          WAIT 1.0
298          CALL Load_disk_data(Tcal(*),Tfilesize,Data_id$,Flg)
300          IF NOT Flg THEN
302             BEEP
304             DISP "NO FILE IN MEMORY, TRY AGAIN."
306             WAIT 1.5
308             GOSUB Reset_filestuff
310             GOTO Tcaldata
312          END IF
314          GOSUB Reset_filestuff
316          ALLOCATE True_t(Datacount)
318          CALL Real_pnt_time(Tcal(*),True_t(*),Tfilesize,Datacount)
320       END IF
322       ALLOCATE Cal_wv(Datacount,2)
324       Intr_prty=Local_prty
```

```
326       IF Ch_vcal$ = "y" OR Ch_vcal$ = "Y" THEN
328          CALL Fix_voltage(Wave(*),Vcal(*),Datacount,Vfilesize)
330       END IF
332       IF Ch_tcal$ = "Y" OR Ch_tcal$ = "y" THEN
334          CALL Fix_time(Wave(*),True_t(*),Cal_wv(*),Datacount,Stp_flg)
336       ELSE
338          MAT Cal_wv(*,1) = Wave(*,1)
340          MAT Cal_wv(*,2) = Wave(*,2)
342       END IF
344       IF NOT Stp_flg THEN
346          INPUT "Enter a 40 character description of the data.",Data_id$
348          Intr_prty = Intr_prty + 2
350          CALL Data_to_disk_r(1,Datacount,Cal_wv(*),Data_id$)
352          Local_prty = Intr_prty
354          PRINT "End of program; type 'RUN' to repeat."
356       END IF
358       SUBEXIT
360 Reset_filestuff:      !
362       Diskdrive$ = ""
364       Filename$ = ""
366       Flg = 0
368       RETURN
370    SUBEND
372    !
374    !*****************************************************************
376    !
378    SUB Real_pnt_time(REAL Tcal(*),True_t(*),INTEGER Tfilesize,Datacount)
380    !
382 Real_pnt_time:   !
384    !
386       OPTION BASE 1
388       RAD
390    !
392       COM /Tcal_vars/ REAL Point,Kount,Scope_window,Real_window
394    !
396    !
398       Point = Tcal(Tfilesize,1)!number of points in the tcal acquisition
400       Kount = Tcal(Tfilesize-1,1)!number of zero crossings found
402       Scope_window = Tcal(Tfilesize-2,1)!10*time per division
404       Real_window = Tcal(Tfilesize-3,1)!measured time window
406    !
408    ! find the "true" time per point for points before the first tcal
410    ! interval.
412    !
414       FOR I = 1 TO INT(Tcal(1,2))
416          True_t(I) = (I-1)*Tcal(1,1)/Tcal(1,2)
418       NEXT I
420    !
422    ! Now the points between the first and last crossing. This is the
424    ! calibrated portion of the time window. All that comes before
426    ! and all that comes after the zero crossings are an estimate of the
428    ! actual time per point.
430    !
432       FOR J = 2 TO Kount
434          FOR I = INT(Tcal(J-1,2)) + 1  TO INT(Tcal(J,2))
436             Xy = I*1.-Tcal(J-1,2)
```

```
438             True_t(I)=Xy*(Tcal(J,1)-Tcal(J-1,1))/(Tcal(J,2)-Tcal(J-1,2))+Tcal(J-1,1)
440          NEXT I
442       NEXT J
444    ! Now the points after the final zero crossing interval.
446       FOR I=INT(Tcal(Kount,2))+1  TO Datacount
448          Xy=1.*I-Tcal(Kount,2)
450          True_t(I)=Xy*(Real_window-Tcal(Kount,1))/(Point-Tcal(Kount,2))+Tcal(Kount,1)
452       NEXT I
454       IF Bug1 THEN
456          FOR I=1 TO Datacount
458             PRINT I;" ";True_t(I)
460          NEXT I
462       END IF
464       SUBEXIT
466    SUBEND
468    !
470    !*****************************************************************************
472    !
474    SUB Fix_voltage(REAL Wv(*),Vc(*),INTEGER Datacount,Lvls)
476    !
478    ! Vc(1,1)= # of voltage levels, Vc(2,1)=null
480    ! Vc(i,1)= measured voltage; 2<=i<=levels+1
482    ! Vc(i,2)= calibration voltage
484    !
486 Fix_voltage:   !
488    !
490       OPTION BASE 1
492       RAD
494    !
496    !
498    !
500       J=2
502       Bug1=0
504       FOR I=1 TO Datacount
506          IF Bug1 THEN PRINT I;" BEFORE ";Wv(I,2);
508          Temp=Wv(I,2)
510          IF Temp<Vc(2,1) THEN
512             Wv(I,2)=Wv(I,2)*(Vc(3,2)-Vc(2,2))/(Vc(3,1)-Vc(2,1))
514          END IF
516          IF Temp>Vc(Lvls,1) THEN
518             Xy=(Wv(I,2)-Vc(Lvls,1))*(Vc(Lvls,2)-Vc(Lvls-1,2))
520             Wv(I,2)=Xy/(Vc(Lvls,1)-Vc(Lvls-1,1))+Vc(Lvls,2)
522          END IF
524          IF (Temp>=Vc(2,1)) AND (Temp<=Vc(Lvls,1)) THEN
526             J=Lvls
528             WHILE (Wv(I,2)<=Vc(J,1))
530                J=J-1
532             END WHILE
534             Xy=(Wv(I,2)-Vc(J,1))
536             A=(Wv(I,2)-Vc(2,1))
538             B=(Vc(J+1,1)-Vc(J,1))
540             C=(Vc(J+1,2)-Vc(J,2))
542             D=(Vc(J,2)-Vc(2,2))
544             Wv(I,2)=Xy*(Vc(J+1,2)-Vc(J,2))/(Vc(J+1,1)-Vc(J,1))+Vc(J,2)
546             Wv(I,2)=Wv(I,2)*(D+Xy*(C/B))/A
548          END IF
```

```
550         IF Bug1 THEN PRINT "AFTER ";Wv(I,2);"VCAL";Vc(J,1)
552       NEXT I
554       Bug1=0
556       SUBEXIT
558     SUBEND
560 !
562 !*************************************************************************
564 !
566     SUB Fix_time(REAL Wv(*),True_t(*),Cal_wv(*),INTEGER Datacount,Stp_flg)
568     !
570 Fix_time:   !
572     !
574       OPTION BASE 1
576       RAD
578     !
580       COM /Tcal_vars/ REAL Point,Kount,Scope_window,Real_window
582       REAL Dt,Extrp
584       INTEGER Opt
586       !
588       Opt=0
590       !
592       Dt=Scope_window/Point
594       !
596       Cal_wv(1,1)=Wv(1,1)
598       Cal_wv(1,2)=Wv(1,2)
600       Cnt=2
602       J=1
604       WHILE Cnt<=Datacount
606         Temp=Dt*(Cnt-1)
608         WHILE (True_t(J)<Temp) AND (J<Datacount)
610           J=J+1
612         END WHILE
614         IF J>Point THEN GOTO Stp_loop
616         Cal_wv(Cnt,1)=Wv(Cnt,1)
618         Cal_wv(Cnt,2)=Wv(J-1,2)+(Wv(J,2)-Wv(J-1,2))*(Temp-True_t(J-1))/(True_t(J)-True_t(J-1))
620         IF Bug1 THEN
622           PRINT Cnt;" BEFORE ";Wv(Cnt,2);
624           PRINT " AFTER ";Cal_wv(Cnt,2)
626         END IF
628         Cnt=Cnt+1
630       END WHILE
632 Stp_loop:      !
634       PRINT Cnt-1;"POINTS CORRECTED"
636       IF Real_window<(Scope_window-Dt) THEN
638         CLEAR SCREEN
640         PRINT "The true time window is less than the scope time window."
642 Select_opt:          !
644         PRINT "Your options are to:"
646         PRINT "(0) Abort"
648         PRINT "(1) Don't correct the time base"
650         PRINT "(2) Correct and output fewer data points"
652         PRINT "(3) Extrapolate using the last data point value"
654         PRINT "(4) Extrapolate using a value input from the keyboard "
656         PRINT "(5) Extrapolate using the mean value of the last 5% of the data"
658         PRINT "(6) Extrapolate using the mean slope of the last 5% of the data"
660         INPUT "Your choice ?",Opt
```

```
662         Five_percent=Cnt-1-INT(.05*Datacount)
664         SELECT Opt
666         CASE 0
668            Stp_flg=1
670         CASE 1
672            FOR I=1 TO Datacount
674               Cal_wv(I,1)=Wv(I,1)
676               Cal_wv(I,2)=Wv(I,2)
678            NEXT I
680         CASE 2
682            Datacount=Cnt-1
684         CASE 3
686            FOR I=Cnt TO Datacount
688               Cal_wv(J,1)=Cal_wv(Cnt-1,1)
690               Cal_wv(J,2)=Cal_wv(Cnt-1,2)
692            NEXT I
694         CASE 4
696            INPUT "Value for extrapolation?",Extrp
698            FOR I=Cnt TO Datacount
700               Cal_wv(I,2)=Extrp
702               Cal_wv(I,1)=Wv(I,1)
704            NEXT I
706         CASE 5
708            Extrp=0.
710            FOR I=Five_percent TO Cnt-1
712               Extrp=Extrp+Cal_wv(I,2)
714            NEXT I
716            Extrp=Extrp/(I-1-Five_percent)
718            FOR I=Cnt TO Datacount
720               Cal_wv(I,2)=Extrp
722               Cal_wv(I,1)=Wv(I,1)
724            NEXT I
726         CASE 6
728            Extrp=(Cal_wv(Cnt-1,2)-Cal_wv(Five_percent,2))/(Cnt-Five_percent-1)
730            FOR I=Cnt TO Datacount
732               Cal_wv(I,2)=Extrp
734               Cal_wv(I,1)=Wv(I,1)
736            NEXT I
738         CASE ELSE
740            DISP "That is not one of your choices. Try again."
742            GOTO Select_opt
744         END SELECT
746      END IF
748      Bug1=0
750      CLEAR SCREEN
752      SUBEXIT
754   SUBEND
756 !
758 !*****************************************************************************
760 !
762   SUB File_menu(Mask$,Ftype$,Fls$(*),INTEGER Fls_cnt,Dir_on,Prt_on)
764 File_menu:   !
766      ! Original: 29 Jun 1987, G. Koepke
768      ! Revision: 02 Dec 1987, 07:00
770      OPTION BASE 1
772      DEG
```

```
774     COM /Sys/ Sys_id$[10]
776     COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
778     COM /Interrupts/ INTEGER Intr_prty
780     COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
782     DIM Directory$(600)[80],Bd$(600)[71]
784     DIM D$[80],T$[51],Ids$[40],Stat$[1],Test$[256]
786     INTEGER Bd_cnt,File_cnt,I,C_cnt,C0(1),Format_error,End_search
788     IF Fls_cnt>0 THEN ALLOCATE INTEGER Choose(Fls_cnt)
790     !
792     ! Catalog the disk specified
794     !
796     End_search=0
798     REPEAT  ! Generate path to file and extract file name.
800        ON ERROR GOTO Cat_errors
802        DISP " Reading the Directory ... "
804        IF LEN(Ms_path$)>0 THEN
806           MASS STORAGE IS Ms_path$[1,LEN(Ms_path$)-1]&Diskdrive$
808        ELSE
810           MASS STORAGE IS Diskdrive$
812        END IF
814        CAT TO Directory$(*);NO HEADER,COUNT File_cnt
816        OFF ERROR
818        !
820        ! set up array of legal file names.
822        !
824        Bd_cnt=0
826        MAT Bd$= ("")
828        FOR I=1 TO File_cnt
830           SELECT Directory$(I)[32,36]
832           CASE Ftype$          ! Ftype$ = "BDAT " or
834                                ! Ftype$ = "PROG "
836              IF LEN(Mask$)>0 THEN ! Test for mask$
838                 IF Directory$(I)[1,LEN(Mask$)]=Mask$ THEN
840                    Bd_cnt=Bd_cnt+1
842                    Bd$(Bd_cnt)=Directory$(I)[1;14]&" - "&Ftype$
844                 END IF
846              ELSE
848                 Bd_cnt=Bd_cnt+1
850                 Bd$(Bd_cnt)=Directory$(I)[1;14]&" - "&Ftype$
852              END IF
854           CASE "DIR  "         ! plus all "DIR  " listings
856              Bd_cnt=Bd_cnt+1
858              Bd$(Bd_cnt)=Directory$(I)[1;14]&" - DIR  "
860           CASE ELSE
862           END SELECT
864        NEXT I
866        IF LEN(Ms_path$)>0 AND Bd_cnt>0 AND Fls_cnt>0 THEN
868           Bd_cnt=Bd_cnt+1
870           Bd$(Bd_cnt)="-----  MOVE back up ONE Directory level."
872           Bd_cnt=Bd_cnt+1
874           Bd$(Bd_cnt)="-----  RETURN to ROOT Directory."
876        END IF
878        !
880        ! set up file menu
882        !
884        D$=" Select "&VAL$(Fls_cnt)&" file name(s) for data entry."
```

```
886       T$="List of "&Ftype$&"files and DIRs on "&Diskdrive$
888       IF LEN(Mask$)>0 THEN
890          T$=T$&" mask="&Mask$
892       END IF
894       IF Bd_cnt>0 THEN
896          IF Dir_on>0 THEN GOSUB Read_data_id
898          IF Prt_on THEN
900             GOSUB List_directory
902             End_search=1
904          ELSE
906             C_cnt=Fls_cnt
908             DISP CHR$(12)
910             IF Fls_cnt>0 THEN
912                CALL Menu_scroll(D$,T$,Bd$(*),Bd_cnt,C_cnt,Choose(*))
914             ELSE
916                CALL Menu_scroll(D$,T$,Bd$(*),Bd_cnt,C_cnt,C0(*))
918             END IF
920             !
922             ! transfer file names to Fls$(*).
924             !
926             IF C_cnt=0 THEN ! selection process aborted
928                End_search=1
930                MAT Fls$= ("")
932             ELSE
934                MAT SORT Choose(*)
936                FOR I=1 TO C_cnt
938                   IF Bd$(Choose(I))[18,22]=Ftype$ THEN
940                      Fls$(I)=Bd$(Choose(I))[1;14]
942                      End_search=1
944                   ELSE  ! it must be a Directory or message.
946                      SELECT Bd$(Choose(I))[18,22]
948                      CASE "up ON" ! move up one directory
950                         LOOP
952                            Ms_path$=Ms_path$[1,LEN(Ms_path$)-1]
954                         EXIT IF LEN(Ms_path$)=0
956                            Test$=Ms_path$[LEN(Ms_path$);1]
958                         EXIT IF Test$="/"
960                         END LOOP
962                      CASE "ROOT "  ! jump to root directory
964                         Ms_path$=""
966                      CASE "DIR "  ! add directory to Ms_path$
968                         Test$=TRIM$(Bd$(Choose(I))[1,14])
970                         Ms_path$=Ms_path$&Test$&"/"
972                      CASE ELSE
974                         DISP "ERROR in directory jump"
976                         PAUSE
978                      END SELECT
980                      I=C_cnt
982                   END IF
984                NEXT I
986             END IF
988          END IF
990       ELSE
992          DISP " This directory contains no ";Ftype$;" files ... "
994          WAIT 2.5
996          End_search=1
```

```
998         END IF
1000        DISP CHR$(12)
1002      UNTIL End_search
1004      SUBEXIT
1006 Cat_errors:!
1008      DISP " ERROR ... ";ERRM$
1010      BEEP
1012      CALL Pause_key_on
1014      DISP CHR$(12)
1016      C_cnt=0
1018      MAT Fls$= ("")
1020      SUBEXIT
1022      !
1024      ! ////////////////////////////////////////////////
1026      !
1028 Read_data_id: ! This routine expects to see Ids$ from
1030            ! GRAPH_DATA raw data files.
1032      DISP " Reading file contents ... Please stand by. "
1034      PRINT TABXY(1,18);" Reading #";
1036      FOR I=1 TO Bd_cnt    ! each BDAT file
1038         PRINT TABXY(11,18);
1040         PRINT USING "3D,4A,3D,2A,#";I," of ",Bd_cnt,". "
1042         Ids$="Data not recognized."
1044         IF Bd$(I)[18,22]="BDAT " THEN
1046            ON ERROR GOTO Not_recognized
1048            ASSIGN @Io_path TO Bd$(I)[1;14]
1050            ENTER @Io_path;Stat$
1052            SELECT Stat$
1054            CASE "N"
1056               ENTER @Io_path;Ids$
1058            CASE "Y"
1060               Ids$="Complete graph in GRAPH_DATA form."
1062            END SELECT
1064 Not_recognized:ASSIGN @Io_path TO *
1066            OFF ERROR
1068            IF Dir_on=2 THEN
1070               GOSUB Interpret_1
1072               IF Format_error THEN GOTO Other_format
1074               GOTO Go_on
1076            END IF
1078 Other_format:!
1080            Bd$(I)[23,71]=" ... "&Ids$
1082         END IF
1084 Go_on:NEXT I
1086      PRINT TABXY(1,18);RPT$(" ",40);
1088      DISP CHR$(12);
1090      RETURN
1092      !
1094      ! ////////////////////////////////////////////////
1096      !
1098 Interpret_1: ! This is used to interpret ID strings.
1100      Format_error=1
1102      ! identify this particular format
1104      RETURN
1106      !
1108      ! ////////////////////////////////////////////////
```

```
1110      !
1112 List_directory: ! This routine will provide a tabular listing of
1114              ! the directory along with lds$ if provided
1116              !
1118      DISP " Listing directory ... "
1120      ON TIMEOUT 7,10 GOTO Printer_kaput
1122      PRINTER IS Printer
1124      PRINT USING "//"
1126      PRINT T$
1128      IF LEN(Ms_path$)>0 THEN PRINT "HFS Path:  ";Ms_path$
1130      PRINT RPT$(" =",80)
1132      PRINT "File name";
1134      IF Dir_on THEN
1136         PRINT "      - TYPE  ... contents"
1138      ELSE
1140         PRINT "      - TYPE"
1142      END IF
1144      PRINT RPT$("-",80)
1146      FOR I=1 TO Bd_cnt
1148         IF Bd$(I)[18,22]=Ftype$ OR Bd$(I)[18,22]="DIR  " THEN
1150            PRINT Bd$(I)
1152         END IF
1154      NEXT I
1156      PRINT RPT$("_",80)
1158      PRINT
1160      PRINTER IS CRT
1162      OFF TIMEOUT 7
1164      RETURN
1166 Printer_kaput:DISP " Printer not responding ... listing aborted. "
1168      BEEP
1170      WAIT 1.8
1172      OFF TIMEOUT 7
1174      RETURN
1176   SUBEND
1178   !
1180   ! ***************************************************************
1182   !
1184   SUB Select_disk
1186 Select_disk: ! Original: 13 Nov 1984
1188              ! Revision: 02 Dec 1987
1190      OPTION BASE 1
1192      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
1194      COM /Interrupts/ INTEGER Intr_prty
1196      COM /Sys_msi/ Msi_id$
1198      COM /Sys/ Sys_id$
1200      INTEGER Local_prty,Dd,Pt,Choose(1)
1202      DIM Disc$(30)[60],Title$[40],Displ$[60]
1204      Local_prty=Intr_prty
1206      OFF KEY
1208      !
1210      ! Define the disk drives available for this system, reserve the
1212      ! first characters for the drive address and the characters after
1214      ! the - for a description of the drive.
1216      !
1218      ! Example:
1220      ! Disc$(1)=":,700,0,0      HP 9133H HARD disk, volume 0."
```

```
1222     !
1224     !
1226     Displ$=" SELECT DISK DRIVE  ... Abort will cancel. "
1228     Title$=" Available disk drives for this system. "
1230     Pt=1   ! allow only one select
1232     !
1234     IF Diskdrive$[1,1]<>":" THEN Diskdrive$=""
1236     IF Msi_id$[1,1]<>":" THEN Msi_id$=SYSTEM$("MSI")
1238     IF Msi_id$[1,1]<>":" THEN ! Must be HFS subdirectory
1240        Ms_path$=Msi_id$[1,POS(Msi_id$,":")-1]  ! strip off subdirs
1242        IF Ms_path$[LEN(Ms_path$);1]<>"/" THEN Ms_path$=Ms_path$&"/"
1244        Msi_id$=Msi_id$[POS(Msi_id$,":"),LEN(Msi_id$)]
1246     END IF
1248     Diskdrive$=TRIM$(Diskdrive$)
1250     Msi_id$=TRIM$(Msi_id$)
1252     IF LEN(Diskdrive$)>0 AND LEN(Msi_id$)>0 THEN
1254        Disc$(1)=Diskdrive$&RPT$(" ",17-LEN(Diskdrive$))
1256        Disc$(1)=Disc$(1)&"- Last selected disk drive."
1258        Dd=1
1260        IF Diskdrive$<>Msi_id$ THEN
1262           Disc$(2)=Msi_id$&RPT$(" ",17-LEN(Msi_id$))
1264           Disc$(2)=Disc$(2)&"- Start-up mass storage unit specifier."
1266           Dd=Dd+1
1268        ELSE
1270           Disc$(1)=Disc$(1)&"  Start-up MSUS."
1272        END IF
1274     ELSE
1276        IF LEN(Msi_id$)>0 THEN
1278           Disc$(1)=Msi_id$&RPT$(" ",17-LEN(Msi_id$))
1280           Disc$(1)=Disc$(1)&"- Start-up mass storage unit specifier."
1282           Dd=1
1284        ELSE
1286           Dd=0
1288        END IF
1290     END IF
1292 Disk: !
1294     ! ..... customize system drives here .....
1296     ! Follow format with - after unit specifier, description is
1298     ! optional but recommended.
1300     ! ..........................................................
1302     !
1304     Disc$(Dd+1)=":,702,0          - HP 9122 dual microfloppy left drive"
1306     Disc$(Dd+2)=":,702,1          - HP 9122 dual microfloppy right drive"
1308     Disc$(Dd+3)=":,703,0          - HP 9125 single 5.25 floppy drive"
1310     Disc$(Dd+4)=":,1400           - HP 9133H hard disk volume 1"
1312     !
1314     Dd=Dd+4    ! add the number of drive specifiers above
1316     !
1318     IF Sys_id$[1,4]<>"S300" THEN
1320        Disc$(Dd+1)=":,4,1            - LEFT internal series 200"
1322        Disc$(Dd+2)=":,4,0            - RIGHT internal series 200"
1324        Dd=Dd+2
1326     END IF
1328     ! ..........................................................
1330     !
1332     CALL Menu_scroll(Displ$,Title$,Disc$(*),Dd,Pt,Choose(*))
```

```
1334      IF Pt=0 THEN
1336         Diskdrive$="NO DISK"
1338      ELSE
1340         Dd=POS(Disc$(Choose(Pt)),"-")-1  ! find -
1342         IF Dd>5 THEN  !  valid msus
1344            Diskdrive$=TRIM$(Disc$(Choose(Pt))[1,Dd])
1346         ELSE
1348            DISP " ERROR in reading MSUS from string, - chr not found. "
1350            BEEP
1352            CALL Pause_key_on
1354            Diskdrive$="NO DISK"
1356         END IF
1358      END IF
1360 Diskselected:OFF KEY
1362      SUBEXIT
1364   SUBEND
1366   !
1368   ! ********************************************************
1370   !
1372   SUB Enterfilename(Ac$)
1374 Enterfilename:  ! Original: 13 Nov 1984
1376              ! Revision: 10 Dec 1990  includes HFS directories
1378      OPTION BASE 1
1380      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
1382      COM /Interrupts/ INTEGER Intr_prty
1384      INTEGER I,Ascii_num,Maskflag,Namelength
1386      DIM Test$[256],Hfs_temp$[161]
1388      Namelength=10
1390      IF LEN(Ms_path$)>0 THEN OUTPUT KBD USING "K,#";"#"&Ms_path$&"H"
1392      DISP " ENTER HFS directory PATH (no file)";
1394      IF Ac$<>"PATH" THEN
1396         DISP ", ENTER / for HFS ROOT or null for LIF...";
1398      END IF
1400      LINPUT Hfs_temp$
1402      Hfs_temp$=TRIM$(Hfs_temp$)
1404      IF LEN(Hfs_temp$)>0 THEN
1406         IF LEN(Hfs_temp$)>1 AND Hfs_temp$[LEN(Hfs_temp$);1]<>"/" THEN
1408            Hfs_temp$=Hfs_temp$&"/"
1410         END IF
1412         IF LEN(Hfs_temp$)=1 THEN Hfs_temp$=""
1414         Namelength=14
1416      END IF
1418      IF Ac$="PATH" THEN
1420         Ms_path$=Hfs_temp$
1422         SUBEXIT
1424      END IF
1426      IF LEN(Filename$)>0 THEN OUTPUT KBD USING "K,#";"#"&Filename$&"H"
1428 Efn: !
1430      DISP " ENTER the FILE NAME ... ";
1432      SELECT Ac$
1434      CASE "CAT"
1436         DISP "(ENTER CAT mask* or ENTER null to CAT)";
1438      CASE "ABORT"
1440         DISP "(ENTER null to ABORT) ";
1442      CASE "VALID"
1444         DISP "(must be a VALID name!) ";
```

```
1446      END SELECT
1448      LINPUT Test$
1450      Test$=TRIM$(Test$)
1452      IF LEN(Test$)=0 AND Ac$="VALID" THEN GOTO Enterfilename
1454      IF LEN(Test$)=0 THEN Abortline
1456      IF LEN(Test$)>Namelength THEN
1458         BEEP
1460         DISP "ERROR in NAME ENTRY - max ";Namelength;" chars, you have ";
1462         DISP LEN(Test$);" "
1464         WAIT 1.8
1466         OUTPUT 2 USING "K,#";"#"&Test$&"H"
1468         GOTO Efn
1470      END IF
1472      IF POS(Test$,"*")>1 THEN
1474         Test$=Test$[1,POS(Test$,"*")-1]
1476         Maskflag=1
1478      ELSE
1480         Maskflag=0
1482      END IF
1484      FOR I=1 TO LEN(Test$)
1486         Ascii_num=NUM(Test$[I])
1488         SELECT Ascii_num
1490         CASE 65 TO 90,95,97 TO 122,48 TO 57
1492            !Allowed characters
1494         CASE ELSE
1496            BEEP
1498            DISP "ERROR in NAME ENTRY--ILLEGAL CHARACTERS, TRY AGAIN."
1500            WAIT 1.8
1502            OUTPUT 2 USING "K,#";"#"&Test$&"H"
1504            GOTO Efn
1506         END SELECT
1508      NEXT I
1510      IF Maskflag THEN
1512         Filename$=Test$&"*"
1514      ELSE
1516         Filename$=Test$
1518      END IF
1520      Ms_path$=Hfs_temp$
1522      SUBEXIT
1524 Abortline:Filename$=""
1526      IF Ac$="CAT" THEN Ms_path$=Hfs_temp$
1528      SUBEXIT
1530   SUBEND
1532   !
1534   ! ************************************************************
1536   !
1538   SUB Menu_scroll(D$,T$,Items$(*),INTEGER Item_cnt,To_select,Choose(*))
1540 Menu_scroll:! Original: 22 Jun 1987, Galen Koepke, NBS 723.04
1542          ! Revision: 22 Aug 1990, 12:00, Dennis Carnell
1544          !
1546          ! A general purpose menu utility for scrolling items and
1548          ! selecting either a fixed number or a random number
1550          ! of items.
1552          !  for fixed  : To_select > 0
1554          !  for random : To_select = -1
1556          ! The items are arranged in screens of 15 items each and
```

```
1558        ! the user may access screens via softkeys. There may be
1560        ! up to 40 screens or 600 items to choose from.
1562        ! Maximum sizes: D$[80], T$[51], Items(*)[70]
1564        ! Items$(*) contains the item descriptions
1566        ! Item_cnt is the number of items in Items$(*)
1568        ! Choose(*) is dimensioned to the number of required choices
1570        !         and will be filled with the item numbers chosen.
1572        ! To_select is the number of required choices.
1574        !
1576     OPTION BASE 1
1578     PRINTER IS CRT
1580     DEG
1582     GOSUB Def_variables
1584     GOSUB Define_screens
1586     GOSUB Make_selections
1588     IF Null_file THEN  ! reset to zero
1590        Item_cnt=0
1592        Items$(1)=""
1594        To_select=0    ! no valid selections
1596     END IF
1598     SUBEXIT
1600     !
1602     ! ////////////////////////////////////////////////
1604     !
1606 Def_variables:!
1608     COM /Interrupts/ INTEGER Intr_prty
1610     COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
1612     COM /Sys/ Sys_id$[10]
1614     !
1616     INTEGER Screen_cnt,Items_per_scn,First_item(40),Last_item(40)
1618     INTEGER I,J,K,First_line,Last_line,Active_screen,Pointer,Last_pt
1620     INTEGER Local_prty,Skips,Knobcount,Pointeractive,K0,Null_file
1622     INTEGER Exit_flag,Temp,Random_select,Indx
1624     DIM Marker$[8],Test$[256]
1626     !
1628     ! initialize parameters
1630     !
1632     Local_prty=Intr_prty
1634     IF Local_prty<1 THEN Local_prty=10
1636     IF LEN(Sys_id$)=0 THEN Sys_id$=SYSTEM$("SYSTEM ID")
1638     IF Item_cnt<1 THEN
1640        Null_file=1
1642        Item_cnt=1
1644        To_select=0
1646        Items$(1)="**** Empty ****"
1648     ELSE
1650        Null_file=0
1652     END IF
1654     IF To_select=-1 THEN
1656        Random_select=1   ! choose random number of items
1658        To_select=0       ! needed for softkeys
1660     END IF
1662     IF To_select>Item_cnt THEN To_select=Item_cnt
1664     MAT Choose= (999)
1666     Skips=0
1668     Knobcount=0
```

```
1670      Doneflag=0
1672      Marker$="===>"&RPT$(CHR$(8),4)
1674      RETURN
1676      !
1678      ! ////////////////////////////////////////////////
1680      !
1682 Define_screens:! Set up screens of 15 items each.
1684            !
1686      Items_per_scn=15    ! Maximum number of displayable items
1688      IF INT(Item_cnt/Items_per_scn)=Item_cnt/Items_per_scn THEN
1690         Screen_cnt=INT(Item_cnt/Items_per_scn)
1692      ELSE
1694         Screen_cnt=INT(Item_cnt/Items_per_scn)+1
1696      END IF
1698      J=1
1700      FOR I=1 TO Screen_cnt   ! set up each screen
1702         First_item(I)=J
1704         IF J+Items_per_scn-1<Item_cnt THEN
1706            Last_item(I)=J+Items_per_scn-1
1708            J=J+Items_per_scn
1710         ELSE
1712            Last_item(I)=Item_cnt
1714         END IF
1716      NEXT I
1718      RETURN
1720      !
1722      ! ////////////////////////////////////////////////
1724      !
1726 Make_selections:! MENU setup and use.
1728      Active_screen=1   ! first screen is active
1730      First_line=2      ! first printed line on screen = 2 or greater.
1732      GOSUB Write_screen  ! activate screen at Active_screen
1734                          ! and set First_line and Last_line for Pointer
1736                          ! write Marker$ to first non-selected line.
1738      K0=0                ! Keys start at zero
1740      Exit_flag=0         ! allow ENTER key to exit when selections filled.
1742 Key_loop:  !
1744      ON KBD,Local_prty GOSUB Process_kbd
1746      ON KNOB .01,Local_prty GOSUB Move_pointer
1748      IF Random_select THEN
1750         ! set keys for random selection
1752         DISP D$
1754         ON KEY K0 LABEL " Select",Local_prty GOSUB Select_random
1756         ON KEY K0+9 LABEL " Accept",Local_prty GOTO Exit_line
1758      ELSE ! set key K0 for fixed selection
1760         IF Skips<To_select THEN
1762            DISP D$
1764            IF To_select>1 THEN
1766               Test$=" Select "&VAL$(Skips+1)&" of "&VAL$(To_select)
1768            ELSE
1770               Test$=" Select"
1772            END IF
1774            ON KEY K0 LABEL Test$,Local_prty GOSUB Select_fixed
1776         ELSE
1778            IF To_select>0 THEN
1780               DISP " Selection process complete ..."
```

```
1782            ELSE
1784               DISP " Menu for information only ... "
1786            END IF
1788            ON KEY K0 LABEL "Accept",Local_prty GOTO Exit_line
1790         END IF
1792      END IF
1794      IF Active_screen<Screen_cnt THEN
1796         ON KEY K0+1 LABEL " Next  Screen",Local_prty GOSUB Next_screen
1798      ELSE
1800         OFF KEY K0+1
1802      END IF
1804      IF Active_screen>1 THEN
1806         ON KEY K0+2 LABEL " Last  Screen",Local_prty GOSUB Last_screen
1808      ELSE
1810         OFF KEY K0+2
1812      END IF
1814      IF Skips>0 OR Random_select THEN
1816         ON KEY K0+3 LABEL " Reset  Select",Local_prty GOSUB Select_reset
1818      ELSE
1820         OFF KEY K0+3
1822      END IF
1824      IF To_select>0 OR Random_select THEN
1826         ON KEY K0+4 LABEL " Abort ",Local_prty GOTO Escape_line
1828      ELSE
1830         OFF KEY K0+4
1832      END IF
1834      IF Screen_cnt>2 THEN
1836         ON KEY K0+6 LABEL "Jump to Screen",Local_prty GOSUB Jump_to_scn
1838      ELSE
1840         OFF KEY K0+6
1842      END IF
1844      IF Exit_flag THEN Exit_line
1846      GOTO Key_loop
1848 Escape_line:Skips=0
1850      MAT Choose= (0)
1852      To_select=0
1854 Exit_line:OFF KEY
1856      MAT SORT Choose(*)
1858      OFF KNOB
1860      OFF KBD
1862      OUTPUT KBD;CHR$(255)&CHR$(75);
1864      PRINT CHR$(128);
1866      ! everything cleared, now go back to work.
1868      RETURN
1870      !
1872      ! ////////////////////////////////////////////////
1874      !
1876 Next_screen:     !
1878      OFF KBD
1880      OFF KNOB
1882      OFF KEY
1884      IF Active_screen=Screen_cnt THEN RETURN
1886      Active_screen=Active_screen+1
1888      GOSUB Write_screen
1890      RETURN
1892      !
```

```
1894      ! ////////////////////////////////////////////////
1896      !
1898 Last_screen:      !
1900      OFF KBD
1902      OFF KNOB
1904      OFF KEY
1906      IF Active_screen=1 THEN RETURN
1908      Active_screen=Active_screen-1
1910      GOSUB Write_screen
1912      RETURN
1914      !
1916      ! ////////////////////////////////////////////////
1918      !
1920 Jump_to_errors:DISP " Not a valid screen number ... try again. "
1922      BEEP
1924      WAIT 1.8
1926 Jump_to_scn: !
1928      OFF KBD
1930      OFF KNOB
1932      OFF KEY
1934      DISP " ENTER the screen number desired (1 to ";Screen_cnt;").";
1936      LINPUT Test$
1938      Test$=TRIM$(Test$)
1940      IF LEN(Test$)=0 THEN Jump_to_return
1942      ON ERROR GOTO Jump_to_errors
1944      Temp=INT(VAL(Test$))
1946      OFF ERROR
1948      IF Temp<1 OR Temp>Screen_cnt THEN Jump_to_errors
1950      Active_screen=Temp
1952      GOSUB Write_screen
1954 Jump_to_return: !
1956      DISP CHR$(12)
1958      Test$=""
1960      RETURN
1962      !
1964      ! ////////////////////////////////////////////////
1966      !
1968 Select_fixed:!
1970      OFF KBD
1972      OFF KNOB
1974      OFF KEY
1976      IF NOT Pointeractive THEN
1978         DISP "NO additional selections for this screen."
1980         BEEP
1982         WAIT 2
1984         DISP CHR$(12);
1986         RETURN
1988      END IF
1990      IF Skips=To_select THEN
1992         IF To_select=0 THEN
1994            DISP "This menu is for information only,";
1996            DISP " no selection allowed."
1998         ELSE
2000            DISP "All selections have been filled,";
2002            DISP " 'Select Reset' to repeat."
2004         END IF
```

```
2006        BEEP
2008        WAIT 2
2010        DISP CHR$(12);
2012        RETURN
2014      END IF
2016      Skips=Skips+1
2018      Choose(Skips)=First_item(Active_screen)+Pointer-First_line
2020      PRINT CHR$(129);  ! inverse video
2022      PRINT TABXY(10,Pointer);Items$(Choose(Skips))
2024      PRINT CHR$(128);
2026      PRINT TABXY(1,Pointer);
2028      SELECT Pointer
2030      CASE First_line
2032        GOSUB Point_forward
2034      CASE Last_line
2036        GOSUB Point_backward
2038      CASE ELSE
2040        ! move forward unless it requires wrapping to beginning.
2042        IF Skips-1>0 THEN  ! check for selected items.
2044          I=Pointer-First_line
2046          LOOP
2048            K=0
2050            FOR J=1 TO Skips
2052              IF First_item(Active_screen)+I=Choose(J)  THEN K=1
2054            NEXT J
2056          EXIT IF K=0
2058            I=I+1
2060            IF I+First_line>Last_line THEN K=-1
2062          EXIT IF K=-1
2064          END LOOP
2066          IF K=0 THEN
2068            GOSUB Point_forward
2070          ELSE
2072            GOSUB Point_backward
2074          END IF
2076        ELSE
2078          GOSUB Point_forward
2080        END IF
2082      END SELECT
2084      RETURN
2086      !
2088      ! ////////////////////////////////////////////////
2090      !
2092  Select_random:!
2094      OFF KBD
2096      OFF KNOB
2098      OFF KEY
2100      Test$="NO"
2102      IF NOT Pointeractive THEN
2104        DISP "NO additional selections for this screen."
2106        BEEP
2108        WAIT 2
2110        DISP CHR$(12);
2112        RETURN
2114      END IF
2116      FOR I=1 TO To_select
```

```
2118        IF Choose(I)=First_item(Active_screen)+Pointer-First_line  THEN
2120           Indx=I
2122           Test$="YES"
2124        END IF
2126     NEXT I
2128     SELECT Test$
2130     CASE "YES"              ! Selected item is tagged ... untag
2132        IF Pointer<>Last_item(Active_screen)+1  AND Pointer<>17 THEN
2134           PRINT CHR$(128);! normal video
2136        ELSE
2138           PRINT CHR$(132);! underline video
2140        END IF
2142        PRINT TABXY(10,Pointer);Items$(Choose(Indx))
2144        FOR I=Indx TO To_select-1
2146           Choose(I)=Choose(I+1)
2148        NEXT I
2150        Choose(To_select)=999
2152        To_select=To_select-1
2154     CASE "NO"               ! Selected item is untagged ... tag it
2156        To_select=To_select+1
2158        Choose(To_select)=First_item(Active_screen)+Pointer-First_line
2160        IF Pointer<>Last_item(Active_screen)+1  AND Pointer<>17 THEN
2162           PRINT CHR$(129);! inverse video
2164        ELSE
2166           PRINT CHR$(133);! inverse video with underline
2168        END IF
2170        PRINT TABXY(10,Pointer);Items$(Choose(To_select))
2172     END SELECT
2174     PRINT CHR$(128);
2176     PRINT TABXY(1,Pointer);
2178     RETURN
2180     !
2182     ! ////////////////////////////////////////////////
2184     !
2186 Select_reset:      !Clear Choose file
2188     OFF KBD
2190     OFF KNOB
2192     OFF KEY
2194     IF Random_select THEN To_select=0
2196     Skips=0
2198     MAT Choose= (999)
2200     GOSUB Write_screen
2202     RETURN
2204     !
2206     ! ////////////////////////////////////////////////
2208     !
2210 Process_kbd:! Allow use of arrows and enter key in addition to soft.
2212     Test$=KBD$
2214     IF LEN(Test$)=1 AND Test$[1,1]<>CHR$(32) THEN
2216        BEEP 80.,.1
2218        RETURN
2220     END IF
2222     IF Test$[1,1]=CHR$(32) THEN GOSUB Point_forward
2224     IF Test$[1,1]<>CHR$(255) THEN RETURN
2226     SELECT Test$[2,2]
2228     CASE CHR$(255)
```

```
2230            ! do nothing
2232      CASE "V","T"
2234         GOSUB Point_forward
2236      CASE "^","W"
2238         GOSUB Point_backward
2240      CASE "E","s","t","&"
2242         IF Random_select THEN
2244            GOSUB Select_random
2246         ELSE
2248            IF Skips<To_select THEN
2250               GOSUB Select_fixed
2252            ELSE
2254            ! exit routine
2256               Exit_flag=1
2258            END IF
2260         END IF
2262      CASE ELSE
2264         BEEP 80.,.1
2266      END SELECT
2268      Test$=""
2270      RETURN
2272      !
2274      ! ////////////////////////////////////////////////
2276      !
2278 Point_forward:Knobcount=5
2280      GOSUB Move_pointer
2282      RETURN
2284 Point_backward:Knobcount=-5
2286      GOSUB Move_pointer
2288      RETURN
2290      !
2292      ! ////////////////////////////////////////////////
2294      !
2296 Jog_pointer:! Move the selection pointer on the active screen.
2298            ! without regard to selected values
2300      IF Knobcount>0 THEN  ! Move forward
2302         Pointer=Pointer+1
2304      ELSE                 ! Move backward
2306         Pointer=Pointer-1
2308      END IF
2310      IF Pointer<First_line THEN Pointer=Last_line
2312      IF Pointer>Last_line THEN Pointer=First_line
2314      RETURN
2316      !
2318      ! ////////////////////////////////////////////////////
2320      !
2322 Move_pointer:! Control pointer to avoid re-selection of items
2324      IF NOT Pointeractive THEN RETURN  ! No selections to be made.
2326      Knobcount=Knobcount+KNOBX-KNOBY
2328      IF ABS(Knobcount)<4 THEN RETURN
2330      Last_pt=Pointer
2332      GOSUB Jog_pointer
2334      IF Skips>0 THEN
2336         LOOP
2338            J=Pointer-First_line
2340            FOR I=1 TO Skips
```

```
2342              IF First_item(Active_screen)+J=Choose(I) THEN J=999
2344            NEXT I
2346            IF J=999 AND Pointer=Last_pt THEN Pointeractive=0
2348          EXIT IF Pointeractive=0
2350            IF J=999 THEN GOSUB Jog_pointer
2352          EXIT IF J<>999
2354          END LOOP
2356        END IF
2358        Knobcount=0
2360        OUTPUT KBD;CHR$(255)&CHR$(84);  ! Bring screen home
2362        IF Last_pt=Last_line THEN PRINT CHR$(132);
2364        PRINT "    ";
2366        IF Pointeractive THEN ! Pointer active
2368          IF Pointer=Last_line THEN
2370            PRINT CHR$(132);
2372          ELSE
2374            PRINT CHR$(128);
2376          END IF
2378          PRINT TABXY(1,Pointer);Marker$;CHR$(128);
2380        END IF
2382        RETURN
2384        !
2386        ! ////////////////////////////////////////////////
2388        !
2390 Write_screen:! Write the screen pointed to by Active_screen
2392        ! home and clear screen
2394        OUTPUT KBD;CHR$(255)&CHR$(84)&CHR$(255)&CHR$(75);
2396        Knobcount=KNOBX-KNOBY   ! Clear knob and keyboard
2398        Knobcount=0
2400        Test$=KBD$
2402        Test$=""
2404        !
2406        PRINT TABXY(1,First_line-1);CHR$(132);" Item #| Screen #";
2408        PRINT USING "#,2D,4A,2D,3A";Active_screen," of ";Screen_cnt;" | "
2410        PRINT T$;RPT$(" ",51-LEN(T$));
2412        PRINT TABXY(80,First_line-1);"|";CHR$(128);
2414        J=0
2416        REPEAT
2418          IF J=Last_item(Active_screen)-First_item(Active_screen) THEN
2420            PRINT CHR$(132);
2422            PRINT TABXY(1,First_line+J);RPT$(" ",80)
2424          ELSE
2426            PRINT CHR$(128);
2428          END IF
2430          PRINT TABXY(5,First_line+J);
2432          PRINT USING "3D,A,#";First_item(Active_screen)+J,"|"
2434          IF Random_select THEN
2436            FOR I=1 TO To_select
2438              IF First_item(Active_screen)+J=Choose(I) THEN
2440                PRINT CHR$(129);
2442              END IF
2444            NEXT I
2446          ELSE
2448            IF Skips>0 THEN ! make this line inverse video
2450              FOR I=1 TO Skips
2452                IF First_item(Active_screen)+J=Choose(I) THEN
```

```
2454                    PRINT CHR$(129);
2456                  END IF
2458                NEXT I
2460              END IF
2462           END IF
2464           PRINT TABXY(10,First_line+J);Items$(First_item(Active_screen)+J)
2466           PRINT TABXY(80,First_line+J);"|";
2468           J=J+1
2470       UNTIL J>=(Last_item(Active_screen)-First_item(Active_screen)+1)
2472       Last_line=Last_item(Active_screen)-First_item(Active_screen)
2474       Last_line=Last_line+First_line
2476       !
2478       ! set marker to first non-selected item.
2480       !
2482       Pointeractive=0
2484       IF To_select>0 OR Random_select THEN Pointeractive=1
2486       IF Skips>0 AND Pointeractive=1 THEN ! find first non-selected item
2488          J=0
2490          LOOP
2492             Pointer=First_line+J
2494             FOR I=1 TO Skips
2496                IF First_item(Active_screen)+J=Choose(I) THEN Pointer=0
2498             NEXT I
2500          EXIT IF Pointer<>0
2502             J=J+1
2504             IF First_line+J>Last_line THEN
2506                Pointeractive=0
2508                Pointer=First_line
2510             END IF
2512          EXIT IF Pointer<>0
2514          END LOOP
2516       ELSE
2518          Pointer=First_line
2520       END IF
2522       IF Pointeractive THEN
2524          IF Pointer=Last_line THEN
2526             PRINT CHR$(132);
2528          ELSE
2530             PRINT CHR$(128);
2532          END IF
2534          PRINT TABXY(1,Pointer);Marker$;CHR$(128);
2536       END IF
2538       RETURN
2540    SUBEND
2542    !
2544    ! ***************************************************************
2546    !
2548    SUB Errortrap
2550 Errortrap: ! Original: 13 Nov 1984
2552           ! Revision: 02 Dec 1987
2554       ! Trap most errors here
2556       OPTION BASE 1
2558       COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
2560       DIM File$[20],Test$[256],What$[20],Ac$[5]
2562       BEEP 400,.6
2564       SELECT ERRN
```

```
2566      CASE 54
2568        DISP "DUPLICATE FILE NAME:  ";Filename$;
2570        DISP "....PURGE old one? (Y/N)";
2572        LINPUT What$
2574        What$=TRIM$(What$)
2576        SELECT What$[1,1]
2578        CASE "Y","y"
2580          PURGE Ms_path$&Filename$&Diskdrive$
2582        CASE ELSE
2584          Ac$="VALID"
2586          CALL Enterfilename(Ac$)
2588        END SELECT
2590      CASE 52,53
2592        DISP "Improper FILE NAME -- ENTER NEW FILE NAME";
2594        OUTPUT 2 USING "#,K,K";"#";Filename$
2596        LINPUT Filename$
2598        Filename$=TRIM$(Filename$)
2600      CASE 56
2602        DISP "FILE:  ";Filename$;" is not on this disk, please insert";
2604        DISP " correct disk"
2606        CALL Pause_key_on
2608      CASE 64
2610        DISP "This disk is full, PLEASE insert clean disk"
2612        CALL Pause_key_on
2614      CASE 56
2616        DISP "DATA INPUT disk must be in drive!!  ";
2618        DISP "...CONTINUE when ready."
2620        CALL Pause_key_on
2622      CASE 72,73,76
2624        DISP Diskdrive$;
2626        DISP " is not available, type correct";
2628        DISP " unit specifier (ie. ':,707,0')."
2630        OUTPUT 2 USING "K,#";Diskdrive$
2632        LINPUT Diskdrive$
2634      CASE 80
2636        DISP "CHECK DISK drive door!"
2638        CALL Pause_key_on
2640      CASE ELSE
2642        DISP ERRM$;" 'CONTINUE' when fixed"
2644        CALL Pause_key_on
2646      END SELECT
2648      DISP CHR$(12)
2650      SUBEXIT
2652    SUBEND
2654    !
2656    ! ************************************************************
2658    !
2660    SUB Data_to_disk_r(INTEGER Curve,Datacount,REAL Basket_file(*),Data_id$)
2662  Data_to_disk_r:  ! Original: 13 Nov 1984
2664                   ! Revision: 02 Dec 1987
2666      !This routine will SAVE data files on the disk in RAW data format.
2668      OPTION BASE 1
2670      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
2672      COM /Interrupts/ INTEGER Intr_prty
2674      INTEGER Local_prty,Diskspace
2676      DIM Ac$[5],Status$[1],Tempfile$[14]
```

```
2678      REAL Dtime
2680      OFF KEY
2682      Local_prty=Intr_prty
2684      Dtime=0.
2686      !
2688      !Select the disk drive for data storage
2690      !
2692 Selectdrive: !
2694      GRAPHICS OFF
2696      OUTPUT 2 USING "#,K";"K"
2698      CALL Select_disk
2700      IF Diskdrive$="NO DISK" THEN GOTO Mistakeline
2702 Choosefilename: !
2704      Ac$="ABORT"
2706      Tempfile$=Filename$
2708      CALL Enterfilename(Ac$)
2710      IF LEN(Filename$)=0 THEN
2712         Filename$=Tempfile$
2714         GOTO Mistakeline
2716      END IF
2718 Send_to_disk: ! Create file and save information.
2720      ON ERROR GOTO Cant_savedata
2722      Diskspace=INT((Datacount*16.0)/256)+2
2724      CREATE BDAT Ms_path$&Filename$&Diskdrive$,Diskspace,256
2726      Dtime=TIMEDATE
2728      DISP " SAVING data for CURVE # ";Curve;". "
2730      Status$="N"
2732      ASSIGN @Datapath TO Ms_path$&Filename$&Diskdrive$
2734      OUTPUT @Datapath;Status$
2736      OUTPUT @Datapath;Data_id$    !40 chrs description if single curve.
2738      OUTPUT @Datapath;Datacount   !number of xy points
2740      OUTPUT @Datapath;Datacount   !size of array (same as above)
2742      OUTPUT @Datapath;Basket_file(*)
2744      ASSIGN @Datapath TO *
2746      OFF ERROR
2748      !
2750 Mistakeline:OFF KEY
2752      LOOP
2754      EXIT IF TIMEDATE-Dtime>1.8
2756      END LOOP
2758      DISP CHR$(12)
2760      OUTPUT 2 USING "#,K";"K"
2762      SUBEXIT
2764      !
2766      ! ////////////////////////////////////////////////////////
2768      !
2770 Cant_savedata:  !
2772      BEEP 500,.6
2774      SELECT ERRN
2776      CASE 72,73,76,78,81,82,90,93
2778         DISP Diskdrive$;" has failed or is not available ";
2780         DISP " ....CONTINUE to try again."
2782         CALL Pause_key_on
2784         Filename$=Tempfile$
2786      CASE 84,85
2788         DISP " This disk is not initialized ";
```

```
2790        DISP " ....CONTINUE to try again."
2792        CALL Pause_key_on
2794        Filename$ = Tempfile$
2796      CASE 55,64
2798        DISP " This disk is full, insert new floppy and/or";
2800        DISP " select new drive ...CONTINUE "
2802        CALL Pause_key_on
2804        Filename$ = Tempfile$
2806      CASE ELSE
2808        CALL Errortrap
2810        GOTO Send_to_disk
2812      END SELECT
2814      GOTO Selectdrive
2816      !
2818    SUBEND
2820    !
2822    ! ****************************************************************
2824    !
2826    SUB Load_disk_data(Basket_file(*),INTEGER Basketsize,Data_id$,INTEGER Flg)
2828  Load_disk_data:  ! Original: 13 Nov 1984
2830                   ! Revision: 02 Dec 1987
2832      !This routine will enter data files from the disk
2834      OPTION BASE 1
2836      !
2838      COM /Sys/ Sys_id$
2840      COM /History/ Status$[1],Time_orgn$[8],Date_orgn$[11]
2842      COM /History/ Time_chng$[8],Date_chng$[11],Description$[160]
2844    !
2846      COM /Labels/ Labels$(30)[60],INTEGER Lbl_count,REAL Lbl_addr(30,6)
2848    !Lbl_addr: x, y, pen, size, LDIR, LORG
2850    !
2852      COM /Data_param/ INTEGER Datacount,Filesize,Curvecount,Roster(17,4)
2854      COM /Data_param/ REAL Sym_size,Symbol$(17)[2],Curve_id$(17)[40]
2856      COM /Data_param/ REAL Xmin_data,Xmax_data
2858      COM /Data_param/ REAL Ymin_data,Ymax_data
2860    !
2862    !Roster: Curve#, Start Addr in File(*), Datacount, and PEN
2864    !Symbol$(i)="" or "Y" => no symbol, connect pts
2866    !Symbol$(i)="*Y" =>  * symbol, connect pts
2868    !Symbol$(i)="*N" =>  * symbol, do not connect pts
2870    !
2872      COM /Background/ Graphtype$[12],Margins$(2)[10],Papersize$[1]
2874      COM /Background/ REAL Pen_speed,INTEGER Backgnd_pen,Auto_time
2876      COM /Background/ INTEGER Auto_file,REAL X_cross_y,Y_cross_x
2878      COM /Background/ Xgrid_tick$[4],INTEGER Xmajor,Xminor
2880      COM /Background/ Ygrid_tick$[4],INTEGER Ymajor,Yminor
2882      COM /Background/ REAL Xmin_graph,Xmax_graph,Ymin_graph,Ymax_graph
2884    !
2886      COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
2888      COM /Interrupts/ INTEGER Intr_prty
2890      COM /Enlarge_file/ INTEGER Overflow
2892      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
2894      COM /Data_stuff/ INTEGER Number,REAL Delta_x,REAL Strt_time
2896    !
2898      INTEGER R,Hold_size,Local_prty,Allocated,Fls_cnt
2900      DIM Ac$[5],Tempfile$[10],Mask$[10],Ftype$[5],Fls$(1)[14]
```

```
2902      REAL Dtime
2904      OFF KEY
2906      Local_prty=Intr_prty
2908      !
2910      !Select the disk drive where the data exists
2912      !
2914      IF Overflow<>0 THEN Overflow=0
2916      Hold_size=0
2918      Dtime=0.
2920      Allocated=0
2922 Selectdrive:     !
2924      IF Diskdrive$="NO DISK" THEN Diskdrive$=""
2926      IF LEN(Diskdrive$)>0 THEN GOTO Choosefilename
2928      GRAPHICS OFF
2930      OUTPUT 2 USING "#,K";"K"
2932      CALL Select_disk
2934      IF Diskdrive$="NO DISK" THEN GOTO Mistakelineset
2936 Choosefilename: !
2938      Tempfile$=Filename$
2940      IF LEN(Filename$)>0 THEN GOTO Bring_in_data
2942      Ac$="CAT"
2944      CALL Enterfilename(Ac$)
2946      IF LEN(Filename$)=0 OR POS(Filename$,"*")>1 THEN
2948         IF POS(Filename$,"*")>1 THEN ! set mask$
2950            Mask$=Filename$[1,POS(Filename$,"*")-1]
2952            Filename$=""
2954         ELSE
2956            Mask$=""! no preselection
2958         END IF
2960         Ftype$="BDAT "  ! examine BDAT files only
2962         Fls_cnt=1  ! select one file
2964         Intr_prty=Local_prty+1
2966         CALL File_menu(Mask$,Ftype$,Fls$(*),Fls_cnt,0,0)
2968         Intr_prty=Local_prty
2970         Filename$=Fls$(1)
2972         IF LEN(Filename$)=0 THEN  ! aborted
2974            Filename$=Tempfile$
2976            GOTO Mistakelineset
2978         END IF
2980      END IF
2982 Bring_in_data: !
2984      !
2986      !Find this file on the disk.
2988      !
2990      ON ERROR GOTO Cant_findfile
2992      ASSIGN @Datapath TO Filename$&Diskdrive$
2994      OFF ERROR
2996      Dtime=TIMEDATE
2998      DISP " LOADING disk file: ";Filename$;" ... ";
3000      ON ERROR GOTO Bad_file
3002      ENTER @Datapath;Status$
3004      OFF ERROR
3006      ON ERROR GOTO Cant_findfile
3008      SELECT Status$
3010      CASE "Y"    ! All graphics/data parameters exist.REN 100,2
3012         DISP " Complete graph. "
```

```
3014        ENTER @Datapath;Time_orgn$,Date_orgn$
3016        ENTER @Datapath;Time_chng$,Date_chng$
3018        ENTER @Datapath;Description$
3020        ENTER @Datapath;Labels$(*),Lbl_count,Lbl_addr(*)
3022        ENTER @Datapath;Curve_id$(*),Symbol$(*)
3024        ENTER @Datapath;Roster(*),Curvecount
3026        ENTER @Datapath;Graphtype$,Margins$(*)
3028        ENTER @Datapath;X_cross_y,Y_cross_x
3030        ENTER @Datapath;Xgrid_tick$,Xmajor,Xminor
3032        ENTER @Datapath;Ygrid_tick$,Ymajor,Yminor
3034        ENTER @Datapath;Xmin_graph,Xmax_graph
3036        ENTER @Datapath;Ymin_graph,Ymax_graph
3038      CASE "N"     ! Only data parameters exist.
3040        DISP " RAW data. "
3042      CASE ELSE
3044 Bad_file: DISP CHR$(12)
3046        DISP "Data file is not recognized, entry aborted.";
3048        DISP " ...continue."
3050        BEEP
3052        PAUSE
3054        OFF ERROR
3056        GOTO Mistakelineset
3058      END SELECT
3060      !
3062      ENTER @Datapath;Data_id$
3064      IF Flg THEN
3066        ENTER @Datapath;Delta_x
3068        ENTER @Datapath;Datacount
3070        Hold_size=Datacount
3072      ELSE
3074        ENTER @Datapath;Datacount
3076        ENTER @Datapath;Hold_size
3078      END IF
3080      IF NOT Allocated THEN
3082        IF Datacount>=1 AND Hold_size>=1 THEN
3084          ALLOCATE Holding_file(Hold_size,2)
3086        ELSE
3088          ALLOCATE Holding_file(1,2)
3090        END IF
3092        Allocated=1
3094      END IF
3096      ENTER @Datapath;Holding_file(*)
3098      ASSIGN @Datapath TO *
3100      OFF ERROR
3102      IF NOT Flg THEN
3104        Delta_x=Holding_file(2,1)-Holding_file(1,1)
3106        Strt_time=Holding_file(1,1)
3108        Flg=1
3110      END IF
3112      IF Datacount=0 THEN Mistakeline
3114      !
3116      !Copy data from Holding_file(*) to Basket_file(*)
3118      !
3120      MAT Basket_file= (0.)
3122      IF Datacount>Basketsize THEN   !Receiving file too small.
3124        Allocated=0
```

```
3126          DEALLOCATE Holding_file(*)
3128          DISP " DATA FILE overflow, new data discarded. ";
3130          DISP " (continue) "
3132          BEEP
3134          PAUSE
3136          IF Status$="Y" THEN
3138             Curvecount=0
3140             MAT Roster= (0)
3142          END IF
3144          Overflow=Hold_size
3146          GOTO Mistakelineset
3148       END IF
3150 Copydatafile:      !
3152       FOR R=1 TO Datacount
3154          Basket_file(R,1)=Holding_file(R,1)
3156          Basket_file(R,2)=Holding_file(R,2)
3158       NEXT R
3160       Basketsize=Datacount
3162       GOTO Mistakeline
3164       !
3166 Mistakelineset:Datacount=0
3168 Mistakeline:OFF KEY
3170       IF Allocated THEN DEALLOCATE Holding_file(*)
3172       LOOP
3174       EXIT IF TIMEDATE-Dtime>1.8
3176       END LOOP
3178       DISP CHR$(12)
3180       OUTPUT 2 USING "#,K";"K"
3182       SUBEXIT
3184       !
3186       ! ////////////////////////////////////////////////////
3188       !
3190 Cant_findfile:      !Error in searching for the file.
3192       BEEP 500,.6
3194       SELECT ERRN
3196       CASE 56
3198          DISP "That file does not exist on this disk  ";
3200       CASE 72,73,76,82
3202          DISP Diskdrive$;" has failed or is not available ";
3204       CASE ELSE
3206          DISP ERRM$;
3208       END SELECT
3210       DISP " ....CONTINUE to try again."
3212       PAUSE
3214       Filename$=""
3216       Diskdrive$=""
3218       GOTO Selectdrive
3220       !
3222    SUBEND
3224    !
3226    ! ***************************************************************************
3228    !
```

```
100!   RE-STORE "PULS_PARAMS:,1400"
102   !
104    COM /Sys/ Sys_id$[10]
106    COM /Sys_msi/ Msi_id$[20]
108    COM /Interrupts/ INTEGER Intr_prty
110   !
112    OUTPUT KBD USING "K,#";"SCRATCH KEYE"        !ERASE SOFT KEYS
114    CONTROL KBD,15;0! sets the color of the soft keys
116    CONTROL KBD,2;1
118   !
120    Intr_prty=1
122    CALL Pulse_params
124   !
126    CLEAR SCREEN
128    OUTPUT KBD USING "K,#";"LOAD KEYE"! restore the typing aid keys
130    PRINT TABXY(1,5);"END of program. So long."
132    MASS STORAGE IS ":,1400"
134   !
136    END
138   !
140   !_____________________________________________________________
142   !
144    SUB Pulse_params
146   !
148 Pulse_params:  !
150       OPTION BASE 1
152       RAD
154       COM /Volt_vals/ Vmin,Vmax,V_last,Volt_z,Volt_100,V_first,Vptp
156       COM /Time_vals/ Tm_10,Tm_20,Tm_50,Tm_80,Tm_90,Tr1_10,Tr1_20
158       COM /Time_vals/ Tr2_10,Tr2_20,Pls_dur,Tm2_10,Tm2_20,Tm2_50,Tm2_80
160       COM /Time_vals/ Tm2_90
162       COM /Misc_int/ INTEGER No_o_bins,His_zero_lev
164       COM /Misc_int/ INTEGER His_100_lev,Maxpoint
166       COM /Misc_real/ Delta_v,Delta_v_prc,Ov,Undr
168       COM /Units/ X_units$[20],Y_units$[20]
170       COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
172       !
174 Dateline:      !
176               !-----------------------------------
178               ! This program written by S.M. Chesnut
180               ! October 12, 1990
182               ! Last revision: May 21,1991
184               !-----------------------------------
186       INTEGER Hist_ary(32767),Pnts,Typ
188       REAL Wave(32767,2)
190       DIM Data_id$[40],Dataname$[10],Ch$[1]
192       Ch$="y"
194       !
196       WHILE (Ch$="Y") OR (Ch$="y")
198          DISP "Enter the name of the file when prompted."
200          WAIT 1.0
202          Diskdrive$=""
204          Filename$=""
206          Pnts=32767
208          REDIM Wave(Pnts,2)
```

```
210         CALL Load_disk_data(Wave(*),Pnts,Data_id$)
212         IF Pnts=0 THEN
214            BEEP
216            DISP "NO FILE WAS READ, type continue to try again."
218            PAUSE
220            Pnts=32767
222         ELSE
224            Dataname$=Filename$
226            INPUT "What are the units of the x axis?",X_units$
228            INPUT "What are the units of the y axis?",Y_units$
230            REDIM Wave(Pnts,2)
232            CALL Mak_histogram(Wave(*),Pnts,Hist_ary(*))
234            CALL Parameters(Pnts,Wave(*),Typ)
236            CALL Write_data(Typ,Dataname$)
238         END IF
240         INPUT "Another file? y/n (default is yes)",Ch$
242      END WHILE
244      SUBEXIT
246   SUBEND
248 !
250 !*************************************************************************
252 !
254   SUB Mak_histogram(REAL Wave(*),INTEGER Pnts,INTEGER Hist_ary(*))
256   !
258 Mak_histogram:   !
260              !
262      OPTION BASE 1
264      RAD
266      COM /Volt_vals/ Vmin,Vmax,V_last,Volt_z,Volt_100,V_first,Vptp
268      COM /Time_vals/ Tm_10,Tm_20,Tm_50,Tm_80,Tm_90,Tr1_10,Tr1_20
270      COM /Time_vals/ Tr2_10,Tr2_20,Pls_dur,Tm2_10,Tm2_20,Tm2_50,Tm2_80
272      COM /Time_vals/ Tm2_90
274      COM /Misc_int/ INTEGER No_o_bins,His_zero_lev
276      COM /Misc_int/ INTEGER His_100_lev,Maxpoint
278      COM /Misc_real/ Delta_v,Delta_v_prc,Ov,Undr
280      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
282      COM /Units/ X_units$[20],Y_units$[20]
284      !
286      INTEGER Indx,Min_bin,Half_bin,I,Level,Zipo,Hundred,Done,Auto
288      DIM Ch$[1]
290      DISP "Calculating the histogram, please wait."
292      Auto=1
294      Done=0
296      Level=0
298      Ch$="n"
300      CALL Vmax_min_ptp(Wave(*),Pnts,Vmax,Vmin,Vptp,Maxpoint)
302      No_o_bins=16384
304      Min_bin=Pnts DIV 100
306      WHILE (Ch$="n") OR (Ch$="N")
308         WHILE NOT Done
310            Delta_v=Vptp/No_o_bins
312            Half_bin=No_o_bins DIV 2
314            MAT Hist_ary= (0)
316            FOR I=1 TO Pnts
318               IF 1<=(32767-(INT((Wave(I,2)-Vmin)/Delta_v))) THEN
320                  Level=1+INT((Wave(I,2)-Vmin)/Delta_v)
322               ELSE
324                  Level=No_o_bins
```

```
326             END IF
328             Hist_ary(Level)=Hist_ary(Level)+1
330          NEXT I
332          Bug1=0
334          IF Bug1 THEN
336             PRINTER IS PRT
338             FOR I=1 TO No_o_bins
340                PRINT Hist_ary(I)
342             NEXT I
344             PRINTER IS CRT
346          END IF
348          Bug1=0
350          His_zero_lev=0
352          His_100_lev=0
354          Zipo=0
356          Hundred=0
358          FOR I=1 TO Half_bin
360             IF Hist_ary(I)>His_zero_lev THEN
362                His_zero_lev=Hist_ary(I)
364                Zipo=I
366             END IF
368          NEXT I
370          I=Half_bin
372          WHILE I<No_o_bins
374             IF Hist_ary(I+1)>His_100_lev THEN
376                His_100_lev=Hist_ary(I+1)
378                Hundred=I+1
380             END IF
382             I=I+1
384          END WHILE
386          IF Auto THEN
388             IF (His_zero_lev<Min_bin) THEN
390                No_o_bins=No_o_bins DIV 2
392                IF No_o_bins<128 THEN
394                   GOSUB Hist_message
396                   Done=1
398                END IF
400             ELSE
402                Done=1
404             END IF
406          ELSE
408             Done=1
410          END IF
412       END WHILE
414       GOSUB Calc_v_prms
416       GOSUB Hist_query
418    END WHILE
420    SUBEXIT
422 Hist_message:        !
424    PRINT "The number of bins in the histogram is less than 128."
426    PRINT "Therefore, the voltage resolution is quite bad and you"
428    PRINT "may find it is unacceptable. Keep this in mind when you"
430    PRINT "are asked if the histogram is an acceptable one."
432    WAIT 2.0
434    RETURN
436 !
```

```
438 Calc_v_prms: !
440       Volt_z=Vmin+Zipo*Delta_v-Delta_v/2.
442       Volt_100=Vmin+Hundred*Delta_v-Delta_v/2.
444       V_first=Wave(1,2)
446       V_last=Wave(Pnts,2)
448       Delta_v_prc=Delta_v*100/Vptp
450       RETURN
452 !
454 Hist_query: !
456 !
458       CLEAR SCREEN
460       PRINT "The first waveform point = ";PROUND(V_first,-4);" ";Y_units$;"."
462       PRINT
464       PRINT " The last waveform point = ";PROUND(V_last,-4);" ";Y_units$;"."
466       PRINT
468       PRINT "The minimum = ";PROUND(Vmin,-4);" ";Y_units$;"."
470       PRINT
472       PRINT "The maximum = ";PROUND(Vmax,-4);" ";Y_units$;"."
474       PRINT
476       PRINT "There were ";No_o_bins;"bins used in the histogram."
478       PRINT
480       PRINT "Each histogram bin is equivalent to";PROUND(Delta_v,-5);" ";Y_units$;","
482       PRINT
484       PRINT "or";PROUND(Delta_v_prc,-5);"% of the waveform peak-to-peak."
486       PRINT
488       PRINT "The 0% level occurs at";PROUND(Volt_z,-4);" ";Y_units$;" with";His_zero_lev;"occurences."
490       PRINT
492       PRINT "The 100% level occurs at";PROUND(Volt_100,-4);" ";Y_units$;"
with";His_100_lev;"occurrences."
494       INPUT "Is this an acceptable histogram?",Ch$
496       IF (Ch$="n") OR (Ch$="N") THEN
498          INPUT "How many histogram bins would you like to use?",No_o_bins
500          Done=0
502          Auto=0
504       END IF
506       CLEAR SCREEN
508       RETURN
510    SUBEND
512 !
514 !*************************************************************************
516 !
518    SUB Vmax_min_ptp(REAL Wave(*),INTEGER Pnts,REAL Vmax,Vmin,Vptp,INTEGER Maxpoint)
520 !
522 Vmax_min_ptp: !
524       INTEGER I
526       Vmax=Wave(1,2)
528       Vmin=Wave(1,2)
530       FOR I=1 TO Pnts
532          IF Wave(I,2)<Vmin THEN Vmin=Wave(I,2)
534          IF Wave(I,2)>Vmax THEN
536             Vmax=Wave(I,2)
538             Maxpoint=I
540          END IF
542       NEXT I
544       Vptp=Vmax-Vmin
546       SUBEXIT
```

```
548   SUBEND
550   !
552   !*************************************************************************
554   !
556   SUB File_menu(Mask$,Ftype$,Fls$(*),INTEGER Fls_cnt,Dir_on,Prt_on)
558 File_menu:  !
560     ! Original: 29 Jun 1987, G. Koepke
562     ! Revision: 06 Aug 1987, 10:00
564     OPTION BASE 1
566     DEG
568     COM /Sys/ Sys_id$[10]
570     COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
572     COM /Interrupts/ INTEGER Intr_prty
574     DIM Directory$(150)[80],Bd$(150)[71]
576     DIM D$[80],T$[52],Ids$[40],Stat$[1]
578     INTEGER Bd_cnt,File_cnt,I,C_cnt,C0(1),Format_error
580     IF Fls_cnt>0 THEN ALLOCATE INTEGER Choose(Fls_cnt)
582     !
584     ! Catalog the disk specified
586     !
588     ON ERROR GOTO Cat_errors
590     DISP " Reading the Directory ... "
592     MASS STORAGE IS Diskdrive$
594     CAT TO Directory$(*);NO HEADER,COUNT File_cnt
596     OFF ERROR
598     !
600     ! set up array of legal file names.
602     !
604     Bd_cnt=0
606     FOR I=1 TO File_cnt
608       IF Directory$(I)[32,36]=Ftype$ THEN  ! Ftype$="BDAT "
610                                    ! Ftype$="PROG "
612         IF LEN(Mask$)>0 THEN ! Test for mask$
614           IF Directory$(I)[1,LEN(Mask$)]=Mask$ THEN
616             Bd_cnt=Bd_cnt+1
618             Bd$(Bd_cnt)=Directory$(I)[1;10]
620           END IF
622         ELSE
624           Bd_cnt=Bd_cnt+1
626           Bd$(Bd_cnt)=Directory$(I)[1;10]
628         END IF
630       END IF
632     NEXT I
634     !
636     ! set up file menu
638     !
640     D$="Select "&VAL$(Fls_cnt)&" file names for data entry."
642     T$="List of "&Ftype$&"files on "&Diskdrive$
644     IF LEN(Mask$)>0 THEN
646       T$=T$&" mask="&Mask$
648     END IF
650     IF Bd_cnt>0 THEN
652       IF Dir_on>0 THEN GOSUB Read_data_id
654       IF Prt_on THEN
656         GOSUB List_directory
658       ELSE
```

```
660             C_cnt=Fls_cnt
662             DISP CHR$(12)
664             IF Fls_cnt>0 THEN
666                CALL Menu_scroll(D$,T$,Bd$(*),Bd_cnt,C_cnt,Choose(*))
668             ELSE
670                CALL Menu_scroll(D$,T$,Bd$(*),Bd_cnt,C_cnt,C0(*))
672             END IF
674             !
676             ! transfer file names to Fls$(*).
678             !
680             IF C_cnt=0 THEN ! selection process aborted
682                MAT Fls$= ("")
684             ELSE
686                MAT SORT Choose(*)
688                FOR I=1 TO C_cnt
690                   Fls$(I)=Bd$(Choose(I))[1;10]
692                NEXT I
694             END IF
696          END IF
698       ELSE
700          DISP " This directory contains no BDAT files ... "
702          WAIT 2.5
704       END IF
706       DISP CHR$(12)
708       SUBEXIT
710 Cat_errors:!
712       DISP "ERROR ... ";ERRM$
714       BEEP
716       PAUSE
718       C_cnt=0
720       MAT Fls$= ("")
722       SUBEXIT
724       !
726       ! ////////////////////////////////////////////////
728       !
730 Read_data_id:  ! This routine expects to see Ids$ from
732             ! GRAPH_DATA raw data files.
734       DISP " Reading file contents ... "
736       FOR I=1 TO Bd_cnt    ! each BDAT file
738          Ids$="Data not recognized."
740          ON ERROR GOTO Not_recognized
742          ASSIGN @Io_path TO Bd$(I)[1;10]
744          ENTER @Io_path;Stat$
746          SELECT Stat$
748          CASE "N"
750             ENTER @Io_path;Ids$
752          CASE "Y"
754             Ids$="Complete graph ... use GRAPH_DATA."
756          END SELECT
758 Not_recognized:ASSIGN @Io_path TO *
760          OFF ERROR
762          IF Dir_on=2 THEN
764             GOSUB Interpret_1
766             IF Format_error THEN GOTO Other_format
768             GOTO Go_on
770          END IF
```

```
772 Other_format:!
774         Bd$(I)[11,71]=" ... "&Ids$
776 Go_on: NEXT I
778       RETURN
780       !
782       ! ////////////////////////////////////////////////
784       !
786 Interpret_1: ! This is used to interpret TEM program ID strings.
788       Format_error=0
790       ! identify this particular format
792       IF LEN(Ids$)<40 THEN
794          Format_error=1
796          RETURN
798       END IF
800       IF Ids$[40]<>"*" THEN
802          Format_error=1
804          RETURN
806       END IF
808       ! make the information readable
810       Bd$(I)[11,15]=" ... "
812       Bd$(I)[16,25]=Ids$[1,10]
814       Bd$(I)[26,32]=", "&Ids$[11,12]&":"&Ids$[13,14]
816       Bd$(I)[33,42]=", "&Ids$[15,16]&" "&Ids$[17,18]&" "&Ids$[19,20]
818       Bd$(I)[43,55]=", "&Ids$[21,27]&" MHz"
820       Bd$(I)[56,65]=", "&Ids$[28,33]&"vm"
822       Bd$(I)[66,71]=","&Ids$[38,39]
824       RETURN
826       !
828       ! ////////////////////////////////////////////////
830       !
832 List_directory:  ! This routine will provide a tabular listing of
834               ! the directory along with Ids$ if provided
836               !
838       DISP " Listing directory ... "
840       PRINTER IS PRT
842       PRINT USING "//"
844       PRINT T$
846       PRINT RPT$("-",80)
848       PRINT "File name";
850       IF Dir_on THEN
852          PRINT "  ...  contents"
854       ELSE
856          PRINT
858       END IF
860       PRINT RPT$("-",80)
862       FOR I=1 TO Bd_cnt
864          PRINT Bd$(I)
866       NEXT I
868       PRINT RPT$("_",80)
870       PRINT
872       PRINTER IS CRT
874       RETURN
876    SUBEND
878    !
880    ! ************************************************************
882    !
```

```
884   SUB Fifty_percent(REAL Wave(*),Wv_50,INTEGER Pnts,Start_pos,Typ)
886   !
888 Fifty_percent:  !
890   !
892      OPTION BASE 1
894      RAD
896   !
898      INTEGER I,Found
900      I=Start_pos
902      Found=0
904      WHILE (NOT Found) AND (I<=Pnts) AND (I>0)
906         IF Wave(I,2)<Wv_50 THEN Found=1
908         I=I-1*Typ
910      END WHILE
912      Start_pos=I+1*Typ
914      SUBEXIT
916   SUBEND
918   !
920   !*************************************************************************
922   !
924   DEF FNPuls_typ(REAL Last,Wv_90)
926 Puls_typ:  !
928      IF (Last>Wv_90) THEN
930         RETURN 1! step like waveform
932      ELSE
934         RETURN -1! impulse like
936      END IF
938   FNEND
940   !
942   !*************************************************************************
944   !
946   SUB Rev_pol(INTEGER Pnts,REAL Wave(*),Wv_10,Wv_90,Wv_amp,Wv_base)
948   !
950 Rev_pol:  !
952   !
954      INTEGER I
956      FOR I=1 TO Pnts
958         Wave(I,2)=Wv_amp+Wv_base-Wave(I,2)
960      NEXT I
962      SUBEXIT
964   SUBEND
966   !
968   !*************************************************************************
970   !
972   SUB Time_o_prcnt(REAL Wave(*),Wv_pct,Pct_t,INTEGER Pnts,Drectn,Strt,Typ)
974   !
976   ! This subroutine assumes all waveforms are positive going, i.e. they
978   ! start at on level and increase to another level. If the original
980   ! waveform had a negative polarity, this was remedied earlier in
982   ! the program.
984   !
986 Time_o_prcnt:  !
988   ! Imports:  wave, an array of data
990   !         wv_pct, any desired percentage level
992   !         pnts, the number of data points
994   !         drectn = 1 if wv_pct >= 50%
```

```
996   !         drectn  = -1 if wv_pct < 50%
998   !         typ  = 1 for a step or the positive slope of an impulse
1000  !         typ  = -1 for the negative slope of an impulse
1002  ! Modifies: strt, the time "position" of the percent level of interest.
1004  !
1006     INTEGER I,Found
1008     REAL Dx
1010     I=Strt
1012     Found=0
1014     Dx=Wave(2,1)-Wave(1,1)
1016     WHILE ((NOT Found) AND (I<=Pnts) AND (I>0))
1018        IF Drectn=Typ THEN
1020           IF Wave(I,2)>Wv_pct THEN
1022              Found=1
1024              IF (Typ=1) THEN I=I-1
1026           ELSE
1028              I=I+Drectn
1030           END IF
1032        ELSE
1034           IF Wave(I,2)<Wv_pct THEN
1036              Found=1
1038              IF (Typ=-1) THEN I=I-1
1040           ELSE
1042              I=I+Drectn
1044           END IF
1046        END IF
1048     END WHILE
1050     Pct_t=I*1.0*Dx+Dx*(Wv_pct-Wave(I,2))/(Wave(I+1,2)-Wave(I,2))
1052     SUBEXIT
1054  SUBEND
1056  !
1058  !*****************************************************************
1060  !
1062  SUB Polarity(INTEGER Pnts,Polar,REAL Wave(*),Wv_10,Wv_90)
1064  !
1066 Polarity:  !
1068     OPTION BASE 1
1070     RAD
1072  ! This subroutine determines the polarity of the waveform.
1074  !
1076  ! Imports: pnts, wave, wv_10, & wv_90
1078  ! Exports: polar  = 1 if positive, -1 if negative.
1080  !
1082     INTEGER I,Found
1084  !
1086     Found=0
1088     I=1
1090     WHILE (NOT Found) AND (I<=Pnts)
1092        IF (Wave(I,2)<Wv_10) THEN
1094           Polar=1
1096           Found=1
1098        END IF
1100        IF (Wave(I,2)>Wv_90) THEN
1102           Polar=-1
1104           Found=1
1106        END IF
```

```
1108        I=I+1
1110      END WHILE
1112      SUBEXIT
1114    SUBEND
1116    !
1118    !*************************************************************************
1120    !
1122    SUB Prcnt(REAL Amp,Top,Baseln,Wv_10,Wv_20,Wv_50,Wv_80,Wv_90,Ov,Und,Vmin,Vmax)
1124    !
1126  Prcnt:  !
1128    !
1130      Wv_10=Amp*.1+Baseln
1132      Wv_20=Amp*.2+Baseln
1134      Wv_50=Amp*.5+Baseln
1136      Wv_80=Amp*.8+Baseln
1138      Wv_90=Amp*.9+Baseln
1140      Und=(Baseln-Vmin)/Amp*100
1142      Ov=(Vmax-Top)/Amp*100
1144      SUBEXIT
1146    SUBEND
1148    !
1150    !*************************************************************************
1152    !
1154    SUB Select_value(REAL Wave_top,Wave_base)
1156    !
1158  Select_value:  !
1160    !
1162    ! This program returns the values for the waveform topline and
1164    ! waveform baseline.
1166    !
1168      OPTION BASE 1
1170      RAD
1172    !
1174      COM /Volt_vals/ Vmin,Vmax,V_last,Volt_z,Volt_100,V_first,Vptp
1176      COM /Time_vals/ Tm_10,Tm_20,Tm_50,Tm_80,Tm_90,Tr1_10,Tr1_20
1178      COM /Time_vals/ Tr2_10,Tr2_20,Pls_dur,Tm2_10,Tm2_20,Tm2_50,Tm2_80
1180      COM /Time_vals/ Tm2_90
1182      COM /Misc_int/ INTEGER No_o_bins,INTEGER His_zero_lev
1184      COM /Misc_int/ INTEGER His_100_lev,Maxpoint
1186      COM /Misc_real/ Delta_v,Delta_v_prc,Ov,Undr
1188    !
1190      INTEGER Defn,Defn_z
1192    !
1194      PRINT "The 0% level occurs at";PROUND(Volt_z,-4);" ";Y_units$;" with";His_zero_lev;"occurences."
1196      PRINT "The 100% level occurs at";PROUND(Volt_100,-4);" ";Y_units$;"
                      with";His_100_lev;"occurences."
1198      PRINT " "
1200      PRINT "0% definition ? 1=voltage of first point"
1202      PRINT "                2=voltage of last point"
1204      PRINT "                3=minimum voltage"
1206      PRINT "                4=histogram 0% voltage"
1208      PRINT "              ELSE =user defined voltage"
1210      PRINT " "
1212      INPUT Defn_z
1214      SELECT Defn_z
1216      CASE 1
```

```
1218        Wave_base=V_first
1220      CASE 2
1222        Wave_base=V_last
1224      CASE 3
1226        Wave_base=Vmin
1228      CASE 4
1230        Wave_base=Volt_z
1232      CASE ELSE
1234        INPUT "Input 0% level",Wave_base
1236      END SELECT
1238      IF Wave_base<Vmin THEN
1240        BEEP
1242        PRINT "WARNING: the 0% level is less than the minimum voltage"
1244        PRINT "of the waveform."
1246      END IF
1248      !
1250      PRINT "100% definition ? 1=voltage of first point"
1252      PRINT "                  2=voltage of last point"
1254      PRINT "                  3=maximum voltage"
1256      PRINT "                  4=histogram 0% voltage"
1258      PRINT "             ELSE =user defined voltage"
1260      PRINT " "
1262      INPUT Defn
1264      SELECT Defn
1266      CASE 1
1268        Wave_top=V_first
1270      CASE 2
1272        Wave_top=V_last
1274      CASE 3
1276        Wave_top=Vmax
1278      CASE 4
1280        Wave_top=Volt_100
1282      CASE ELSE
1284        INPUT "Input 100% level",Wave_top
1286      END SELECT
1288      IF Wave_top>Vmax THEN
1290        BEEP
1292        PRINT "WARNING: the 100% level is greater than the maximum"
1294        PRINT "voltage of the waveform."
1296      END IF
1298      Volt_z=Wave_base
1300      Volt_100=Wave_top
1302      CLEAR SCREEN
1304      SUBEXIT
1306    SUBEND
1308    !
1310    !*************************************************************************
1312    !
1314    SUB Parameters(INTEGER Pnts,REAL Wave(*),INTEGER Typ)
1316    !
1318  Parameters:  !
1320          !
1322      OPTION BASE 1
1324      RAD
1326      COM /Volt_vals/ Vmin,Vmax,V_last,Volt_z,Volt_100,V_first,Vptp
1328      COM /Time_vals/ Tm_10,Tm_20,Tm_50,Tm_80,Tm_90,Tr1_10,Tr1_20
```

```
1330     COM /Time_vals/ Tr2_10,Tr2_20,Pls_dur,Tm2_10,Tm2_20,Tm2_50,Tm2_80
1332     COM /Time_vals/ Tm2_90
1334     COM /Misc_int/ INTEGER No_o_bins,His_zero_lev
1336     COM /Misc_int/ INTEGER His_100_lev,Maxpoint
1338     COM /Misc_real/ Delta_v,Delta_v_prc,Ov,Undr
1340   !
1342     INTEGER Start_pos,Polar
1344     REAL Wv_amp,Wv_top,Wv_base,Wv_10,Wv_20,Wv_50,Wv_80,Wv_90
1346   !
1348     Typ=1
1350     CALL Select_value(Wv_top,Wv_base)
1352     Wv_amp=Wv_top-Wv_base
1354     CALL Prcnt(Wv_amp,Wv_top,Wv_base,Wv_10,Wv_20,Wv_50,Wv_80,Wv_90,Ov,Undr,Vmin,Vmax)
1356     !
1358     CALL Polarity(Pnts,Polar,Wave(*),Wv_10,Wv_90)
1360     IF Polar=-1 THEN
1362        CALL Rev_pol(Pnts,Wave(*),Wv_10,Wv_90,Wv_amp,Wv_base)
1364     END IF
1366     !
1368     Start_pos=Maxpoint
1370     CALL Fifty_percent(Wave(*),Wv_50,Pnts,Start_pos,Typ)
1372     !
1374     CALL Time_o_prcnt(Wave(*),Wv_50,Tm_50,Pnts,1,Start_pos,Typ)
1376     CALL Time_o_prcnt(Wave(*),Wv_10,Tm_10,Pnts,-1,Start_pos,Typ)
1378     CALL Time_o_prcnt(Wave(*),Wv_20,Tm_20,Pnts,-1,Start_pos,Typ)
1380     CALL Time_o_prcnt(Wave(*),Wv_80,Tm_80,Pnts,1,Start_pos,Typ)
1382     CALL Time_o_prcnt(Wave(*),Wv_90,Tm_90,Pnts,1,Start_pos,Typ)
1384     !
1386     Tr1_10=Tm_90-Tm_10
1388     Tr1_20=Tm_80-Tm_20
1390     !
1392     Typ=FNPuls_typ(V_last,Wv_90)
1394     IF Typ=-1 THEN
1396        PRINT "This is an impulse-like waveform."
1398        Start_pos=Maxpoint
1400        CALL Fifty_percent(Wave(*),Wv_50,Pnts,Start_pos,Typ)
1402        CALL Time_o_prcnt(Wave(*),Wv_50,Tm2_50,Pnts,1,Start_pos,Typ)
1404        CALL Time_o_prcnt(Wave(*),Wv_10,Tm2_10,Pnts,1,Start_pos,Typ)
1406        CALL Time_o_prcnt(Wave(*),Wv_20,Tm2_20,Pnts,1,Start_pos,Typ)
1408        CALL Time_o_prcnt(Wave(*),Wv_80,Tm2_80,Pnts,-1,Start_pos,Typ)
1410        CALL Time_o_prcnt(Wave(*),Wv_90,Tm2_90,Pnts,-1,Start_pos,Typ)
1412        Tr2_20=Tm2_20-Tm2_80
1414        Tr2_10=Tm2_10-Tm2_90
1416        Pls_dur=Tm2_50-Tm_50
1418     END IF
1420     !
1422     SUBEXIT
1424   SUBEND
1426   !
1428   !*************************************************************************
1430   !
1432   SUB Write_data(INTEGER Typ,Dataname$)
1434   !
1436 Write_data:  !
1438   !
1440     OPTION BASE 1
```

```
1442      RAD
1444    !
1446      COM /Sys/ Sys_id$[10]
1448      COM /Sys_msi/ Msi_id$[20]
1450      COM /Interrupts/ INTEGER Intr_prty
1452      COM /Volt_vals/ Vmin,Vmax,V_last,Volt_z,Volt_100,V_first,Vptp
1454      COM /Time_vals/ Tm_10,Tm_20,Tm_50,Tm_80,Tm_90,Tr1_10,Tr1_20
1456      COM /Time_vals/ Tr2_10,Tr2_20,Pls_dur,Tm2_10,Tm2_20,Tm2_50,Tm2_80
1458      COM /Time_vals/ Tm2_90
1460      COM /Misc_int/ INTEGER No_o_bins,INTEGER His_zero_lev
1462      COM /Misc_int/ INTEGER His_100_lev,Maxpoint
1464      COM /Misc_real/ Delta_v,Delta_v_prc,Ov,Undr
1466      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
1468      COM /Units/ X_units$[20],Y_units$[20]
1470      !
1472      INTEGER Local_prty,Diskspace
1474      DIM Ac$[5],Status$[1],Data_id$[40]
1476      REAL Dtime
1478      OFF KEY
1480      Local_prty=Intr_prty
1482      Dtime=0.
1484      Underline$=CHR$(132)
1486      Enhance_off$=CHR$(128)
1488      !
1490      Filesize=22 ! This can be increased to accommodate more output.
1492      ALLOCATE File$(Filesize)[80]
1494      INPUT "Enter a 40 character description of the data",Data_id$
1496      !Select the disk drive for data storage
1498      !
1500 Selectdrive: !
1502      IF Diskdrive$="NO DISK" THEN Diskdrive$=""
1504      IF LEN(Diskdrive$)>0 THEN GOTO Choosefilename
1506      GRAPHICS OFF
1508      OUTPUT 2 USING "#,K";"K"
1510      CALL Select_disk
1512      IF Diskdrive$="NO DISK" THEN GOTO Mistakeline
1514 Choosefilename: !
1516      IF LEN(Filename$)>0 THEN GOTO Send_to_disk
1518      Ac$="ABORT"
1520      CALL Enterfilename(Ac$)
1522      IF LEN(Filename$)=0 THEN GOTO Mistakeline
1524 Send_to_disk: ! Create file and save information.
1526      ON ERROR GOTO Cant_savedata
1528      Diskspace=INT((82)*Filesize/256)
1530      ! diskspace = (# of characters in a line * # of lines in the file)/
1532      ! the # bytes in a record; this gives # of records needed.
1534      CREATE ASCII Filename$&Diskdrive$,Diskspace
1536      Dtime=TIMEDATE
1538      DISP " SAVING data in file ";Filename$;" on ";Diskdrive$
1540      ASSIGN @Datapath TO Filename$&Diskdrive$
1542      OUTPUT @Datapath;Data_id$    !40 chrs description of data
1544      OUTPUT @Datapath;Filesize    !number of data strings
1546      GOSUB Fill_file
1548      OUTPUT @Datapath;File$(*)
1550      ASSIGN @Datapath TO *
1552      OFF ERROR
```

```
1554      !
1556 Mistakeline:OFF KEY
1558      LOOP
1560      EXIT IF TIMEDATE-Dtime>1.8
1562      END LOOP
1564      DISP CHR$(12)
1566      OUTPUT 2 USING "#,K";"K"
1568      DEALLOCATE File$(*)
1570      SUBEXIT
1572      !
1574      ! ////////////////////////////////////////////////////
1576      !
1578 Cant_savedata:  !
1580      BEEP 500,.6
1582      SELECT ERRN
1584      CASE 72,73,76,78,81,82,90,93
1586         DISP Diskdrive$;" has failed or is not available ";
1588         DISP " ....CONTINUE to try again."
1590         PAUSE
1592         Diskdrive$="*"
1594      CASE 84,85
1596         DISP " This disk is not initialized ";
1598         DISP " ....CONTINUE to try again."
1600         PAUSE
1602         Diskdrive$="*"
1604      CASE 55,64
1606         DISP " This disk is full, insert new floppy and/or";
1608         DISP " select new drive ...CONTINUE "
1610         PAUSE
1612         Diskdrive$="*"
1614      CASE ELSE
1616         CALL Errortrap
1618         IF LEN(Filename$)>0 THEN GOTO Send_to_disk
1620      END SELECT
1622      GOTO Selectdrive
1624 Fill_file:      !
1626      File$(1)="The data file is "&Dataname$&"."
1628      File$(2)="This file is called "&Filename$&"."
1630      File$(3)="The 10%-90% first transition duration is "&VAL$(PROUND(Tr1_10,-13))&"
               "&X_units$&"."
1632      File$(4)="The 20%-80% first transition duration is "&VAL$(PROUND(Tr1_20,-13))&"
               "&X_units$&"."
1634      IF (Typ=-1) THEN
1636         File$(5)="The 10%-90% second transition duration is "&VAL$(PROUND(Tr2_10,-3))&"
               "&X_units$&"."
1638         File$(6)="The 20%-80% second transition duration is "&VAL$(PROUND(Tr2_20,-3))&"
               "&X_units$&"."
1640         File$(7)="The pulse duration is "&VAL$(PROUND(Pls_dur,-3))&"  "&X_units$&"."
1642      ELSE
1644         File$(5)="*"
1646         File$(6)="*"
1648         File$(7)="*"
1650      END IF
1652      File$(8)="The percentage overshoot is "&VAL$(PROUND(Ov,-2))&"%."
1654      File$(9)="The percentage undershoot is "&VAL$(PROUND(Undr,-2))&"%."
1656      File$(10)="The waveform amplitude is "&VAL$(PROUND((Volt_100-Volt_z),-3))&"  "&Y_units$&"."
```

```
1658      File$(11)="The maximum level is "&VAL$(PROUND(Vmax,-3))&" "&Y_units$&"."
1660      File$(12)="The minimum level is "&VAL$(PROUND(Vmin,-3))&" "&Y_units$&"."
1662      File$(13)="There were "&VAL$(No_o_bins)&" bins used in the histogram."
1664      File$(14)="Each histogram bin is equivalent to "&VAL$(PROUND(Delta_v,-6))&" "&Y_units$&"."
1666      File$(15)="or "&VAL$(PROUND(Delta_v_prc,-3))&"% of peak to peak."
1668      File$(16)="The 0% histogram level is "&VAL$(PROUND(Volt_z,-3))&" "&Y_units$
1670      File$(17)="with "&VAL$(His_zero_lev)&" data occurences."
1672      File$(18)="The 100% histogram level is "&VAL$(PROUND(Volt_100,-3))&" "&Y_units$
1674      File$(19)="with "&VAL$(His_100_lev)&" data occurences."
1676      File$(20)="The peak to peak value is "&VAL$(PROUND(Vptp,-3))&" "&Y_units$
1678      File$(21)="The first point in the waveform is "&VAL$(PROUND(V_first,-3))&" "&Y_units$&"."
1680      File$(22)="the last point in the waveform is "&VAL$(PROUND(V_last,-3))&" "&Y_units$&"."
1682      INPUT "Would you like a print out of the data now?",Ch$
1684      IF (Ch$="y") OR (Ch$="Y") THEN
1686         PRINTER IS PRT
1688         FOR I=1 TO Filesize
1690            PRINT File$(I)
1692            PRINT ""
1694         NEXT I
1696         PRINTER IS CRT
1698      END IF
1700      RETURN
1702 !
1704   SUBEND
1706 !
1708 !*****************************************************************************
1710 !
1712   SUB Enterfilename(Ac$)
1714 Enterfilename:  ! Original: 13 Nov 1984
1716               ! Revision: 10 Dec 1990  includes HFS directories
1718      OPTION BASE 1
1720      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
1722      COM /Interrupts/ INTEGER Intr_prty
1724      INTEGER I,Ascii_num,Maskflag,Namelength
1726      DIM Test$[256],Hfs_temp$[161]
1728      Namelength=10
1730      IF LEN(Ms_path$)>0 THEN OUTPUT KBD USING "K,#";"#"&Ms_path$&"H"
1732      DISP " ENTER HFS directory PATH (no file)";
1734      IF Ac$<>"PATH" THEN
1736         DISP ", ENTER / for HFS ROOT or null for LIF...";
1738      END IF
1740      LINPUT Hfs_temp$
1742      Hfs_temp$=TRIM$(Hfs_temp$)
1744      IF LEN(Hfs_temp$)>0 THEN
1746         IF LEN(Hfs_temp$)>1 AND Hfs_temp$[LEN(Hfs_temp$);1]<>"/" THEN
1748            Hfs_temp$=Hfs_temp$&"/"
1750         END IF
1752         IF LEN(Hfs_temp$)=1 THEN Hfs_temp$=""
1754         Namelength=14
1756      END IF
1758      IF Ac$="PATH" THEN
1760         Ms_path$=Hfs_temp$
1762         SUBEXIT
1764      END IF
1766      IF LEN(Filename$)>0 THEN OUTPUT KBD USING "K,#";"#"&Filename$&"H"
1768 Efn: !
```

```
1770      DISP " ENTER the FILE NAME ... ";
1772      SELECT Ac$
1774      CASE "CAT"
1776         DISP "(ENTER CAT mask* or ENTER null to CAT)";
1778      CASE "ABORT"
1780         DISP "(ENTER null to ABORT) ";
1782      CASE "VALID"
1784         DISP "(must be a VALID name!) ";
1786      END SELECT
1788      LINPUT Test$
1790      Test$=TRIM$(Test$)
1792      IF LEN(Test$)=0 AND Ac$="VALID" THEN GOTO Enterfilename
1794      IF LEN(Test$)=0 THEN Abortline
1796      IF LEN(Test$)>Namelength THEN
1798         BEEP
1800         DISP "ERROR in NAME ENTRY - max ";Namelength;" chars, you have ";
1802         DISP LEN(Test$);" "
1804         WAIT 1.8
1806         OUTPUT 2 USING "K,#";"#"&Test$&"H"
1808         GOTO Efn
1810      END IF
1812      IF POS(Test$,"*")>1 THEN
1814         Test$=Test$[1,POS(Test$,"*")-1]
1816         Maskflag=1
1818      ELSE
1820         Maskflag=0
1822      END IF
1824      FOR I=1 TO LEN(Test$)
1826         Ascii_num=NUM(Test$[I])
1828         SELECT Ascii_num
1830         CASE 65 TO 90,95,97 TO 122,48 TO 57
1832            !Allowed characters
1834         CASE ELSE
1836            BEEP
1838            DISP "ERROR in NAME ENTRY--ILLEGAL CHARACTERS, TRY AGAIN."
1840            WAIT 1.8
1842            OUTPUT 2 USING "K,#";"#"&Test$&"H"
1844            GOTO Efn
1846         END SELECT
1848      NEXT I
1850      IF Maskflag THEN
1852         Filename$=Test$&"*"
1854      ELSE
1856         Filename$=Test$
1858      END IF
1860      Ms_path$=Hfs_temp$
1862      SUBEXIT
1864 Abortline:Filename$=""
1866      IF Ac$="CAT" THEN Ms_path$=Hfs_temp$
1868      SUBEXIT
1870   SUBEND
1872   !
1874   !*****************************************************************
1876   !
1878   SUB Select_disk
1880 Select_disk: ! Original: 13 Nov 1984
```

```
1882          ! Revision: 02 Dec 1987
1884      OPTION BASE 1
1886      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
1888      COM /Interrupts/ INTEGER Intr_prty
1890      COM /Sys_msi/ Msi_id$
1892      COM /Sys/ Sys_id$
1894      INTEGER Local_prty,Dd,Pt,Choose(1)
1896      DIM Disc$(30)[60],Title$[40],Displ$[60]
1898      Local_prty=Intr_prty
1900      OFF KEY
1902      !
1904      ! Define the disk drives available for this system, reserve the
1906      ! first characters for the drive address and the characters after
1908      ! the - for a description of the drive.
1910      !
1912      ! Example:
1914      ! Disc$(1)=":,700,0,0      HP 9133H HARD disk, volume 0."
1916      !
1918      !
1920      Displ$=" SELECT DISK DRIVE  ... Abort will cancel. "
1922      Title$=" Available disk drives for this system. "
1924      Pt=1   ! allow only one select
1926      !
1928      IF Diskdrive$[1,1]<>":" THEN Diskdrive$=""
1930      IF Msi_id$[1,1]<>":" THEN Msi_id$=SYSTEM$("MSI")
1932      IF Msi_id$[1,1]<>":" THEN ! Must be HFS subdirectory
1934         Ms_path$=Msi_id$[1,POS(Msi_id$,":")-1]  ! strip off subdirs
1936         IF Ms_path$[LEN(Ms_path$);1]<>"/" THEN Ms_path$=Ms_path$&"/"
1938         Msi_id$=Msi_id$[POS(Msi_id$,":"),LEN(Msi_id$)]
1940      END IF
1942      Diskdrive$=TRIM$(Diskdrive$)
1944      Msi_id$=TRIM$(Msi_id$)
1946      IF LEN(Diskdrive$)>0 AND LEN(Msi_id$)>0 THEN
1948         Disc$(1)=Diskdrive$&RPT$(" ",17-LEN(Diskdrive$))
1950         Disc$(1)=Disc$(1)&"- Last selected disk drive."
1952         Dd=1
1954         IF Diskdrive$<>Msi_id$ THEN
1956            Disc$(2)=Msi_id$&RPT$(" ",17-LEN(Msi_id$))
1958            Disc$(2)=Disc$(2)&"- Start-up mass storage unit specifier."
1960            Dd=Dd+1
1962         ELSE
1964            Disc$(1)=Disc$(1)&"  Start-up MSUS."
1966         END IF
1968      ELSE
1970         IF LEN(Msi_id$)>0 THEN
1972            Disc$(1)=Msi_id$&RPT$(" ",17-LEN(Msi_id$))
1974            Disc$(1)=Disc$(1)&"- Start-up mass storage unit specifier."
1976            Dd=1
1978         ELSE
1980            Dd=0
1982         END IF
1984      END IF
1986 Disk: !
1988      ! ..............  customize system drives here .....................
1990      ! Follow format with - after unit specifier, description is
1992      ! optional but recommended.
```

```
1994      ! ..........................................................
1996      !
1998      Disc$(Dd+1)=":,702,0          - HP 9122 dual microfloppy left drive"
2000      Disc$(Dd+2)=":,702,1          - HP 9122 dual microfloppy right drive"
2002      Disc$(Dd+3)=":,703,0          - HP 9125 single 5.25 floppy drive"
2004      Disc$(Dd+4)=":,1400           - HP 9133H hard disk volume 1"
2006      !
2008      Dd=Dd+4    ! add the number of drive specifiers above
2010      !
2012      IF Sys_id$[1,4]<>"S300" THEN
2014         Disc$(Dd+1)=":,4,1           - LEFT internal series 200"
2016         Disc$(Dd+2)=":,4,0           - RIGHT internal series 200"
2018         Dd=Dd+2
2020      END IF
2022      ! ..........................................................
2024      !
2026      CALL Menu_scroll(Displ$,Title$,Disc$(*),Dd,Pt,Choose(*))
2028      IF Pt=0 THEN
2030         Diskdrive$="NO DISK"
2032      ELSE
2034         Dd=POS(Disc$(Choose(Pt)),"-")-1  ! find -
2036         IF Dd>5 THEN  ! valid msus
2038            Diskdrive$=TRIM$(Disc$(Choose(Pt))[1,Dd])
2040         ELSE
2042            DISP " ERROR in reading MSUS from string, - chr not found. "
2044            BEEP
2046            CALL Pause_key_on
2048            Diskdrive$="NO DISK"
2050         END IF
2052      END IF
2054 Diskselected:OFF KEY
2056      SUBEXIT
2058   SUBEND
2060   !
2062   ! ************************************************************
2064   !
2066   SUB File_menu(Mask$,Ftype$,Fls$(*),INTEGER Fls_cnt,Dir_on,Prt_on)
2068 File_menu:  !
2070      ! Original: 29 Jun 1987, G. Koepke
2072      ! Revision: 02 Dec 1987, 07:00
2074      OPTION BASE 1
2076      DEG
2078      COM /Sys/ Sys_id$[10]
2080      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
2082      COM /Interrupts/ INTEGER Intr_prty
2084      COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
2086      DIM Directory$(600)[80],Bd$(600)[71]
2088      DIM D$[80],T$[51],Ids$[40],Stat$[1],Test$[256]
2090      INTEGER Bd_cnt,File_cnt,I,C_cnt,C0(1),Format_error,End_search
2092      IF Fls_cnt>0 THEN ALLOCATE INTEGER Choose(Fls_cnt)
2094      !
2096      ! Catalog the disk specified
2098      !
2100      End_search=0
2102      REPEAT   ! Generate path to file and extract file name.
2104         ON ERROR GOTO Cat_errors
```

```
2106        DISP " Reading the Directory ... "
2108        IF LEN(Ms_path$)>0 THEN
2110           MASS STORAGE IS Ms_path$[1,LEN(Ms_path$)-1]&Diskdrive$
2112        ELSE
2114           MASS STORAGE IS Diskdrive$
2116        END IF
2118        CAT TO Directory$(*);NO HEADER,COUNT File_cnt
2120        OFF ERROR
2122        !
2124        ! set up array of legal file names.
2126        !
2128        Bd_cnt=0
2130        MAT Bd$= ("")
2132        FOR I=1 TO File_cnt
2134           SELECT Directory$(I)[32,36]
2136           CASE Ftype$          ! Ftype$ = "BDAT " or
2138                                ! Ftype$ = "PROG "
2140              IF LEN(Mask$)>0 THEN ! Test for mask$
2142                 IF Directory$(I)[1,LEN(Mask$)]=Mask$ THEN
2144                    Bd_cnt=Bd_cnt+1
2146                    Bd$(Bd_cnt)=Directory$(I)[1;14]&" - "&Ftype$
2148                 END IF
2150              ELSE
2152                 Bd_cnt=Bd_cnt+1
2154                 Bd$(Bd_cnt)=Directory$(I)[1;14]&" - "&Ftype$
2156              END IF
2158           CASE "DIR "          ! plus all "DIR " listings
2160              Bd_cnt=Bd_cnt+1
2162              Bd$(Bd_cnt)=Directory$(I)[1;14]&" - DIR "
2164           CASE ELSE
2166           END SELECT
2168        NEXT I
2170        IF LEN(Ms_path$)>0 AND Bd_cnt>0 AND Fls_cnt>0 THEN
2172           Bd_cnt=Bd_cnt+1
2174           Bd$(Bd_cnt)="----  MOVE back up ONE Directory level."
2176           Bd_cnt=Bd_cnt+1
2178           Bd$(Bd_cnt)="----  RETURN to ROOT Directory."
2180        END IF
2182        !
2184        ! set up file menu
2186        !
2188        D$=" Select "&VAL$(Fls_cnt)&" file name(s) for data entry."
2190        T$="List of "&Ftype$&"files and DIRs on "&Diskdrive$
2192        IF LEN(Mask$)>0 THEN
2194           T$=T$&" mask="&Mask$
2196        END IF
2198        IF Bd_cnt>0 THEN
2200           IF Dir_on>0 THEN GOSUB Read_data_id
2202           IF Prt_on THEN
2204              GOSUB List_directory
2206              End_search=1
2208           ELSE
2210              C_cnt=Fls_cnt
2212              DISP CHR$(12)
2214              IF Fls_cnt>0 THEN
2216                 CALL Menu_scroll(D$,T$,Bd$(*),Bd_cnt,C_cnt,Choose(*))
```

```
2218            ELSE
2220               CALL Menu_scroll(D$,T$,Bd$(*),Bd_cnt,C_cnt,C0(*))
2222            END IF
2224            !
2226            ! transfer file names to Fls$(*).
2228            !
2230            IF C_cnt=0 THEN ! selection process aborted
2232               End_search=1
2234               MAT Fls$= ("")
2236            ELSE
2238               MAT SORT Choose(*)
2240               FOR I=1 TO C_cnt
2242                  IF Bd$(Choose(I))[18,22]=Ftype$ THEN
2244                     Fls$(I)=Bd$(Choose(I))[1;14]
2246                     End_search=1
2248                  ELSE  ! it must be a Directory or message.
2250                     SELECT Bd$(Choose(I))[18,22]
2252                     CASE "up ON" ! move up one directory
2254                        LOOP
2256                           Ms_path$=Ms_path$[1,LEN(Ms_path$)-1]
2258                        EXIT IF LEN(Ms_path$)=0
2260                           Test$=Ms_path$[LEN(Ms_path$);1]
2262                        EXIT IF Test$="/"
2264                        END LOOP
2266                     CASE "ROOT "  ! jump to root directory
2268                        Ms_path$=""
2270                     CASE "DIR  "  ! add directory to Ms_path$
2272                        Test$=TRIM$(Bd$(Choose(I))[1,14])
2274                        Ms_path$=Ms_path$&Test$&"/"
2276                     CASE ELSE
2278                        DISP "ERROR in directory jump"
2280                        PAUSE
2282                     END SELECT
2284                     I=C_cnt
2286                  END IF
2288               NEXT I
2290            END IF
2292         END IF
2294      ELSE
2296         DISP " This directory contains no ";Ftype$;" files ... "
2298         WAIT 2.5
2300         End_search=1
2302      END IF
2304      DISP CHR$(12)
2306   UNTIL End_search
2308   SUBEXIT
2310 Cat_errors:!
2312   DISP " ERROR ... ";ERRM$
2314   BEEP
2316   CALL Pause_key_on
2318   DISP CHR$(12)
2320   C_cnt=0
2322   MAT Fls$= ("")
2324   SUBEXIT
2326   !
2328   ! ////////////////////////////////////////////////
```

```
2330      !
2332 Read_data_id: ! This routine expects to see Ids$ from
2334            ! GRAPH_DATA raw data files.
2336      DISP " Reading file contents ... Please stand by. "
2338      PRINT TABXY(1,18);" Reading #";
2340      FOR I=1 TO Bd_cnt    ! each BDAT file
2342         PRINT TABXY(11,18);
2344         PRINT USING "3D,4A,3D,2A,#";I," of ",Bd_cnt,". "
2346         Ids$="Data not recognized."
2348         IF Bd$(I)[18,22]="BDAT " THEN
2350            ON ERROR GOTO Not_recognized
2352            ASSIGN @Io_path TO Bd$(I)[1;14]
2354            ENTER @Io_path;Stat$
2356            SELECT Stat$
2358            CASE "N"
2360               ENTER @Io_path;Ids$
2362            CASE "Y"
2364               Ids$="Complete graph in GRAPH_DATA form."
2366            END SELECT
2368 Not_recognized:ASSIGN @Io_path TO *
2370            OFF ERROR
2372            IF Dir_on=2 THEN
2374               GOSUB Interpret_1
2376               IF Format_error THEN GOTO Other_format
2378               GOTO Go_on
2380            END IF
2382 Other_format:!
2384            Bd$(I)[23,71]=" ... "&Ids$
2386         END IF
2388 Go_on:NEXT I
2390      PRINT TABXY(1,18);RPT$(" ",40);
2392      DISP CHR$(12);
2394      RETURN
2396      !
2398      ! ////////////////////////////////////////////////
2400      !
2402 Interpret_1: ! This is used to interpret ID strings.
2404      Format_error=1
2406      ! identify this particular format
2408      RETURN
2410      !
2412      ! ////////////////////////////////////////////////
2414      !
2416 List_directory: ! This routine will provide a tabular listing of
2418               ! the directory along with Ids$ if provided
2420               !
2422      DISP " Listing directory ... "
2424      ON TIMEOUT 7,10 GOTO Printer_kaput
2426      PRINTER IS Printer
2428      PRINT USING "//"
2430      PRINT T$
2432      IF LEN(Ms_path$)>0 THEN PRINT "HFS Path:  ";Ms_path$
2434      PRINT RPT$("=",80)
2436      PRINT "File name";
2438      IF Dir_on THEN
2440         PRINT "      - TYPE  ... contents"
```

```
2442      ELSE
2444         PRINT "      - TYPE"
2446      END IF
2448      PRINT RPT$("-",80)
2450      FOR I=1 TO Bd_cnt
2452         IF Bd$(I)[18,22]=Ftype$ OR Bd$(I)[18,22]="DIR  " THEN
2454            PRINT Bd$(I)
2456         END IF
2458      NEXT I
2460      PRINT RPT$("_",80)
2462      PRINT
2464      PRINTER IS CRT
2466      OFF TIMEOUT 7
2468      RETURN
2470 Printer_kaput:DISP " Printer not responding ... listing aborted. "
2472      BEEP
2474      WAIT 1.8
2476      OFF TIMEOUT 7
2478      RETURN
2480   SUBEND
2482   !
2484   ! ********************************************************************
2486   !
2488   SUB Menu_scroll(D$,T$,Items$(*),INTEGER Item_cnt,To_select,Choose(*))
2490 Menu_scroll:! Original: 22 Jun 1987, Galen Koepke, NBS 723.04
2492           ! Revision: 22 Aug 1990, 12:00, Dennis Camell
2494           !
2496           ! A general purpose menu utility for scrolling items and
2498           ! selecting either a fixed number or a random number
2500           ! of items.
2502           !  for fixed  : To_select > 0
2504           !  for random : To_select = -1
2506           ! The items are arranged in screens of 15 items each and
2508           ! the user may access screens via softkeys. There may be
2510           ! up to 40 screens or 600 items to choose from.
2512           ! Maximum sizes: D$[80], T$[51], Items(*)[70]
2514           ! Items$(*) contains the item descriptions
2516           ! Item_cnt is the number of items in Items$(*)
2518           ! Choose(*) is dimensioned to the number of required choices
2520           !          and will be filled with the item numbers chosen.
2522           ! To_select is the number of required choices.
2524           !
2526      OPTION BASE 1
2528      PRINTER IS CRT
2530      DEG
2532      GOSUB Def_variables
2534      GOSUB Define_screens
2536      GOSUB Make_selections
2538      IF Null_file THEN  ! reset to zero
2540         Item_cnt=0
2542         Items$(1)=""
2544         To_select=0    ! no valid selections
2546      END IF
2548      SUBEXIT
2550      !
2552      ! ////////////////////////////////////////////////////
```

```
2554      !
2556 Def_variables:!
2558      COM /Interrupts/ INTEGER Intr_prty
2560      COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
2562      COM /Sys/ Sys_id$[10]
2564      !
2566      INTEGER Screen_cnt,Items_per_scn,First_item(40),Last_item(40)
2568      INTEGER I,J,K,First_line,Last_line,Active_screen,Pointer,Last_pt
2570      INTEGER Local_prty,Skips,Knobcount,Pointeractive,K0,Null_file
2572      INTEGER Exit_flag,Temp,Random_select,Indx
2574      DIM Marker$[8],Test$[256]
2576      !
2578      ! initialize parameters
2580      !
2582      Local_prty=Intr_prty
2584      IF Local_prty<1 THEN Local_prty=10
2586      IF LEN(Sys_id$)=0 THEN Sys_id$=SYSTEM$("SYSTEM ID")
2588      IF Item_cnt<1 THEN
2590         Null_file=1
2592         Item_cnt=1
2594         To_select=0
2596         Items$(1)="**** Empty ****"
2598      ELSE
2600         Null_file=0
2602      END IF
2604      IF To_select=-1 THEN
2606         Random_select=1   ! choose random number of items
2608         To_select=0       ! needed for softkeys
2610      END IF
2612      IF To_select>Item_cnt THEN To_select=Item_cnt
2614      MAT Choose= (999)
2616      Skips=0
2618      Knobcount=0
2620      Doneflag=0
2622      Marker$="===>"&RPT$(CHR$(8),4)
2624      RETURN
2626      !
2628      ! ////////////////////////////////////////////////
2630      !
2632 Define_screens:! Set up screens of 15 items each.
2634            !
2636      Items_per_scn=15    ! Maximum number of displayable items
2638      IF INT(Item_cnt/Items_per_scn)=Item_cnt/Items_per_scn  THEN
2640         Screen_cnt=INT(Item_cnt/Items_per_scn)
2642      ELSE
2644         Screen_cnt=INT(Item_cnt/Items_per_scn)+1
2646      END IF
2648      J=1
2650      FOR I=1 TO Screen_cnt   ! set up each screen
2652         First_item(I)=J
2654         IF J+Items_per_scn-1<Item_cnt THEN
2656            Last_item(I)=J+Items_per_scn-1
2658            J=J+Items_per_scn
2660         ELSE
2662            Last_item(I)=Item_cnt
2664         END IF
```

```
2666      NEXT I
2668      RETURN
2670      !
2672      ! ////////////////////////////////////////////////
2674      !
2676 Make_selections:! MENU setup and use.
2678      Active_screen=1    ! first screen is active
2680      First_line=2       ! first printed line on screen  =  2 or greater.
2682      GOSUB Write_screen  ! activate screen at Active_screen
2684                          ! and set First_line and Last_line for Pointer
2686                          ! write Marker$ to first non-selected line.
2688      K0=0                  ! Keys start at zero
2690      Exit_flag=0           ! allow ENTER key to exit when selections filled.
2692 Key_loop:  !
2694      ON KBD,Local_prty GOSUB Process_kbd
2696      ON KNOB .01,Local_prty GOSUB Move_pointer
2698      IF Random_select THEN
2700         ! set keys for random selection
2702         DISP D$
2704         ON KEY K0 LABEL " Select",Local_prty GOSUB Select_random
2706         ON KEY K0+9 LABEL " Accept",Local_prty GOTO Exit_line
2708      ELSE ! set key K0 for fixed selection
2710         IF Skips<To_select THEN
2712            DISP D$
2714            IF To_select>1 THEN
2716               Test$=" Select "&VAL$(Skips+1)&" of "&VAL$(To_select)
2718            ELSE
2720               Test$=" Select"
2722            END IF
2724            ON KEY K0 LABEL Test$,Local_prty GOSUB Select_fixed
2726         ELSE
2728            IF To_select>0 THEN
2730               DISP " Selection process complete ..."
2732            ELSE
2734               DISP " Menu for information only ... "
2736            END IF
2738            ON KEY K0 LABEL "Accept",Local_prty GOTO Exit_line
2740         END IF
2742      END IF
2744      IF Active_screen<Screen_cnt THEN
2746         ON KEY K0+1 LABEL " Next   Screen",Local_prty GOSUB Next_screen
2748      ELSE
2750         OFF KEY K0+1
2752      END IF
2754      IF Active_screen>1 THEN
2756         ON KEY K0+2 LABEL " Last   Screen",Local_prty GOSUB Last_screen
2758      ELSE
2760         OFF KEY K0+2
2762      END IF
2764      IF Skips>0 OR Random_select THEN
2766         ON KEY K0+3 LABEL " Reset  Select",Local_prty GOSUB Select_reset
2768      ELSE
2770         OFF KEY K0+3
2772      END IF
2774      IF To_select>0 OR Random_select THEN
2776         ON KEY K0+4 LABEL " Abort ",Local_prty GOTO Escape_line
```

```
2778      ELSE
2780         OFF KEY K0+4
2782      END IF
2784      IF Screen_cnt>2 THEN
2786         ON KEY K0+6 LABEL "Jump to Screen",Local_prty GOSUB Jump_to_scn
2788      ELSE
2790         OFF KEY K0+6
2792      END IF
2794      IF Exit_flag THEN Exit_line
2796      GOTO Key_loop
2798 Escape_line:Skips=0
2800      MAT Choose= (0)
2802      To_select=0
2804 Exit_line:OFF KEY
2806      MAT SORT Choose(*)
2808      OFF KNOB
2810      OFF KBD
2812      OUTPUT KBD;CHR$(255)&CHR$(75);
2814      PRINT CHR$(128);
2816      ! everything cleared, now go back to work.
2818      RETURN
2820      !
2822      ! ////////////////////////////////////////////////
2824      !
2826 Next_screen:      !
2828      OFF KBD
2830      OFF KNOB
2832      OFF KEY
2834      IF Active_screen=Screen_cnt THEN RETURN
2836      Active_screen=Active_screen+1
2838      GOSUB Write_screen
2840      RETURN
2842      !
2844      ! ////////////////////////////////////////////////
2846      !
2848 Last_screen:      !
2850      OFF KBD
2852      OFF KNOB
2854      OFF KEY
2856      IF Active_screen=1 THEN RETURN
2858      Active_screen=Active_screen-1
2860      GOSUB Write_screen
2862      RETURN
2864      !
2866      ! ////////////////////////////////////////////////
2868      !
2870 Jump_to_errors:DISP " Not a valid screen number ... try again. "
2872      BEEP
2874      WAIT 1.8
2876 Jump_to_scn: !
2878      OFF KBD
2880      OFF KNOB
2882      OFF KEY
2884      DISP " ENTER the screen number desired (1 to ";Screen_cnt;").";
2886      LINPUT Test$
2888      Test$=TRIM$(Test$)
```

```
2890      IF LEN(Test$)=0 THEN Jump_to_return
2892      ON ERROR GOTO Jump_to_errors
2894      Temp=INT(VAL(Test$))
2896      OFF ERROR
2898      IF Temp<1 OR Temp>Screen_cnt THEN Jump_to_errors
2900      Active_screen=Temp
2902      GOSUB Write_screen
2904 Jump_to_return: !
2906      DISP CHR$(12)
2908      Test$=""
2910      RETURN
2912      !
2914      ! ////////////////////////////////////////////////
2916      !
2918 Select_fixed:!
2920      OFF KBD
2922      OFF KNOB
2924      OFF KEY
2926      IF NOT Pointeractive THEN
2928         DISP "NO additional selections for this screen."
2930         BEEP
2932         WAIT 2
2934         DISP CHR$(12);
2936         RETURN
2938      END IF
2940      IF Skips=To_select THEN
2942         IF To_select=0 THEN
2944            DISP "This menu is for information only,";
2946            DISP " no selection allowed."
2948         ELSE
2950            DISP "All selections have been filled,";
2952            DISP " 'Select Reset' to repeat."
2954         END IF
2956         BEEP
2958         WAIT 2
2960         DISP CHR$(12);
2962         RETURN
2964      END IF
2966      Skips=Skips+1
2968      Choose(Skips)=First_item(Active_screen)+Pointer-First_line
2970      PRINT CHR$(129); ! inverse video
2972      PRINT TABXY(10,Pointer);Items$(Choose(Skips))
2974      PRINT CHR$(128);
2976      PRINT TABXY(1,Pointer);
2978      SELECT Pointer
2980      CASE First_line
2982         GOSUB Point_forward
2984      CASE Last_line
2986         GOSUB Point_backward
2988      CASE ELSE
2990         ! move forward unless it requires wrapping to beginning.
2992         IF Skips-1>0 THEN  ! check for selected items.
2994            I=Pointer-First_line
2996            LOOP
2998               K=0
3000               FOR J=1 TO Skips
```

```
3002                IF First_item(Active_screen)+I=Choose(J) THEN K=1
3004              NEXT J
3006            EXIT IF K=0
3008              I=I+1
3010              IF I+First_line>Last_line THEN K=-1
3012            EXIT IF K=-1
3014            END LOOP
3016            IF K=0 THEN
3018              GOSUB Point_forward
3020            ELSE
3022              GOSUB Point_backward
3024            END IF
3026          ELSE
3028            GOSUB Point_forward
3030          END IF
3032        END SELECT
3034        RETURN
3036        !
3038        ! ////////////////////////////////////////////////
3040        !
3042 Select_random:!
3044        OFF KBD
3046        OFF KNOB
3048        OFF KEY
3050        Test$="NO"
3052        IF NOT Pointeractive THEN
3054          DISP "NO additional selections for this screen."
3056          BEEP
3058          WAIT 2
3060          DISP CHR$(12);
3062          RETURN
3064        END IF
3066        FOR I=1 TO To_select
3068          IF Choose(I)=First_item(Active_screen)+Pointer-First_line THEN
3070            Indx=I
3072            Test$="YES"
3074          END IF
3076        NEXT I
3078        SELECT Test$
3080        CASE "YES"              ! Selected item is tagged ... untag
3082          IF Pointer<>Last_item(Active_screen)+1 AND Pointer<>17 THEN
3084            PRINT CHR$(128);! normal video
3086          ELSE
3088            PRINT CHR$(132);! underline video
3090          END IF
3092          PRINT TABXY(10,Pointer);Items$(Choose(Indx))
3094          FOR I=Indx TO To_select-1
3096            Choose(I)=Choose(I+1)
3098          NEXT I
3100          Choose(To_select)=999
3102          To_select=To_select-1
3104        CASE "NO"               ! Selected item is untagged ... tag it
3106          To_select=To_select+1
3108          Choose(To_select)=First_item(Active_screen)+Pointer-First_line
3110          IF Pointer<>Last_item(Active_screen)+1 AND Pointer<>17 THEN
3112            PRINT CHR$(129);! inverse video
```

```
3114        ELSE
3116           PRINT CHR$(133);! inverse video with underline
3118        END IF
3120        PRINT TABXY(10,Pointer);Items$(Choose(To_select))
3122     END SELECT
3124     PRINT CHR$(128);
3126     PRINT TABXY(1,Pointer);
3128     RETURN
3130     !
3132     ! ////////////////////////////////////////////////
3134     !
3136 Select_reset:     !Clear Choose file
3138     OFF KBD
3140     OFF KNOB
3142     OFF KEY
3144     IF Random_select THEN To_select=0
3146     Skips=0
3148     MAT Choose= (999)
3150     GOSUB Write_screen
3152     RETURN
3154     !
3156     ! ////////////////////////////////////////////////
3158     !
3160 Process_kbd:! Allow use of arrows and enter key in addition to soft.
3162     Test$=KBD$
3164     IF LEN(Test$)=1 AND Test$[1,1]<>CHR$(32) THEN
3166        BEEP 80.,.1
3168        RETURN
3170     END IF
3172     IF Test$[1,1]=CHR$(32) THEN GOSUB Point_forward
3174     IF Test$[1,1]<>CHR$(255) THEN RETURN
3176     SELECT Test$[2,2]
3178     CASE CHR$(255)
3180           ! do nothing
3182     CASE "V","T"
3184        GOSUB Point_forward
3186     CASE "^","W"
3188        GOSUB Point_backward
3190     CASE "E","s","t","&"
3192        IF Random_select THEN
3194           GOSUB Select_random
3196        ELSE
3198           IF Skips<To_select THEN
3200              GOSUB Select_fixed
3202           ELSE
3204           ! exit routine
3206              Exit_flag=1
3208           END IF
3210        END IF
3212     CASE ELSE
3214        BEEP 80.,.1
3216     END SELECT
3218     Test$=""
3220     RETURN
3222     !
3224     ! ////////////////////////////////////////////////
```

```
3226     !
3228 Point_forward:Knobcount=5
3230     GOSUB Move_pointer
3232     RETURN
3234 Point_backward:Knobcount=-5
3236     GOSUB Move_pointer
3238     RETURN
3240     !
3242     ! ////////////////////////////////////////////////
3244     !
3246 Jog_pointer:! Move the selection pointer on the active screen.
3248          ! without regard to selected values
3250     IF Knobcount>0 THEN  ! Move forward
3252        Pointer=Pointer+1
3254     ELSE                 ! Move backward
3256        Pointer=Pointer-1
3258     END IF
3260     IF Pointer<First_line THEN Pointer=Last_line
3262     IF Pointer>Last_line THEN Pointer=First_line
3264     RETURN
3266     !
3268     ! ////////////////////////////////////////////////////
3270     !
3272 Move_pointer:! Control pointer to avoid re-selection of items
3274     IF NOT Pointeractive THEN RETURN  ! No selections to be made.
3276     Knobcount=Knobcount+KNOBX-KNOBY
3278     IF ABS(Knobcount)<4 THEN RETURN
3280     Last_pt=Pointer
3282     GOSUB Jog_pointer
3284     IF Skips>0 THEN
3286        LOOP
3288           J=Pointer-First_line
3290           FOR I=1 TO Skips
3292              IF First_item(Active_screen)+J=Choose(I) THEN J=999
3294           NEXT I
3296           IF J=999 AND Pointer=Last_pt THEN Pointeractive=0
3298        EXIT IF Pointeractive=0
3300           IF J=999 THEN GOSUB Jog_pointer
3302        EXIT IF J<>999
3304        END LOOP
3306     END IF
3308     Knobcount=0
3310     OUTPUT KBD;CHR$(255)&CHR$(84);  ! Bring screen home
3312     IF Last_pt=Last_line THEN PRINT CHR$(132);
3314     PRINT "    ";
3316     IF Pointeractive THEN ! Pointer active
3318        IF Pointer=Last_line THEN
3320           PRINT CHR$(132);
3322        ELSE
3324           PRINT CHR$(128);
3326        END IF
3328        PRINT TABXY(1,Pointer);Marker$;CHR$(128);
3330     END IF
3332     RETURN
3334     !
3336     ! ////////////////////////////////////////////////
```

```
3338     !
3340 Write_screen:! Write the screen pointed to by Active_screen
3342     ! home and clear screen
3344     OUTPUT KBD;CHR$(255)&CHR$(84)&CHR$(255)&CHR$(75);
3346     Knobcount=KNOBX-KNOBY   ! Clear knob and keyboard
3348     Knobcount=0
3350     Test$=KBD$
3352     Test$=""
3354     !
3356     PRINT TABXY(1,First_line-1);CHR$(132);" Item #| Screen #";
3358     PRINT USING "#,2D,4A,2D,3A";Active_screen," of ";Screen_cnt;" | "
3360     PRINT T$;RPT$(" ",51-LEN(T$));
3362     PRINT TABXY(80,First_line-1);"|";CHR$(128);
3364     J=0
3366     REPEAT
3368        IF J=Last_item(Active_screen)-First_item(Active_screen) THEN
3370           PRINT CHR$(132);
3372           PRINT TABXY(1,First_line+J);RPT$(" ",80)
3374        ELSE
3376           PRINT CHR$(128);
3378        END IF
3380        PRINT TABXY(5,First_line+J);
3382        PRINT USING "3D,A,#";First_item(Active_screen)+J,"|"
3384        IF Random_select THEN
3386           FOR I=1 TO To_select
3388              IF First_item(Active_screen)+J=Choose(I) THEN
3390                 PRINT CHR$(129);
3392              END IF
3394           NEXT I
3396        ELSE
3398           IF Skips>0 THEN ! make this line inverse video
3400              FOR I=1 TO Skips
3402                 IF First_item(Active_screen)+J=Choose(I) THEN
3404                    PRINT CHR$(129);
3406                 END IF
3408              NEXT I
3410           END IF
3412        END IF
3414        PRINT TABXY(10,First_line+J);Items$(First_item(Active_screen)+J)
3416        PRINT TABXY(80,First_line+J);"|";
3418        J=J+1
3420     UNTIL J>=(Last_item(Active_screen)-First_item(Active_screen)+1)
3422     Last_line=Last_item(Active_screen)-First_item(Active_screen)
3424     Last_line=Last_line+First_line
3426     !
3428     ! set marker to first non-selected item.
3430     !
3432     Pointeractive=0
3434     IF To_select>0 OR Random_select THEN Pointeractive=1
3436     IF Skips>0 AND Pointeractive=1 THEN ! find first non-selected item
3438        J=0
3440        LOOP
3442           Pointer=First_line+J
3444           FOR I=1 TO Skips
3446              IF First_item(Active_screen)+J=Choose(I) THEN Pointer=0
3448           NEXT I
```

```
3450        EXIT IF Pointer<>0
3452          J=J+1
3454          IF First_line+J>Last_line THEN
3456            Pointeractive=0
3458            Pointer=First_line
3460          END IF
3462        EXIT IF Pointer<>0
3464        END LOOP
3466      ELSE
3468        Pointer=First_line
3470      END IF
3472      IF Pointeractive THEN
3474        IF Pointer=Last_line THEN
3476          PRINT CHR$(132);
3478        ELSE
3480          PRINT CHR$(128);
3482        END IF
3484        PRINT TABXY(1,Pointer);Marker$;CHR$(128);
3486      END IF
3488      RETURN
3490    SUBEND
3492    !
3494    ! ************************************************************
3496    !
3498    SUB Errortrap
3500  Errortrap: ! Original: 13 Nov 1984
3502           ! Revision: 02 Dec 1987
3504      ! Trap most errors here
3506      OPTION BASE 1
3508      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
3510      DIM File$[20],Test$[256],What$[20],Ac$[5]
3512      BEEP 400,.6
3514      SELECT ERRN
3516      CASE 54
3518        DISP "DUPLICATE FILE NAME:  ";Filename$;
3520        DISP "....PURGE old one? (Y/N)";
3522        LINPUT What$
3524        What$=TRIM$(What$)
3526        SELECT What$[1,1]
3528        CASE "Y","y"
3530          PURGE Ms_path$&Filename$&Diskdrive$
3532        CASE ELSE
3534          Ac$="VALID"
3536          CALL Enterfilename(Ac$)
3538        END SELECT
3540      CASE 52,53
3542        DISP "Improper FILE NAME --- ENTER NEW FILE NAME";
3544        OUTPUT 2 USING "#,K,K";"#";Filename$
3546        LINPUT Filename$
3548        Filename$=TRIM$(Filename$)
3550      CASE 56
3552        DISP "FILE:  ";Filename$;" is not on this disk, please insert";
3554        DISP " correct disk"
3556        CALL Pause_key_on
3558      CASE 64
3560        DISP "This disk is full, PLEASE insert clean disk"
```

```
3562        CALL Pause_key_on
3564      CASE 56
3566        DISP "DATA INPUT disk must be in drive!!  ";
3568        DISP "...CONTINUE when ready."
3570        CALL Pause_key_on
3572      CASE 72,73,76
3574        DISP Diskdrive$;
3576        DISP " is not available, type correct";
3578        DISP " unit specifier (ie. ':,707,0')."";
3580        OUTPUT 2 USING "K,#";Diskdrive$
3582        LINPUT Diskdrive$
3584      CASE 80
3586        DISP "CHECK DISK drive door!"
3588        CALL Pause_key_on
3590      CASE ELSE
3592        DISP ERRM$;" 'CONTINUE' when fixed"
3594        CALL Pause_key_on
3596      END SELECT
3598      DISP CHR$(12)
3600      SUBEXIT
3602    SUBEND
3604    !
3606    ! ************************************************************************
3608    !
3610    SUB Load_disk_data(Basket_file(*),INTEGER Basketsize,Data_id$)
3612  Load_disk_data:  ! Original: 13 Nov 1984
3614                   ! Revision: 02 Dec 1987
3616      !This routine will enter data files from the disk
3618      OPTION BASE 1
3620      !
3622      COM /Sys/ Sys_id$
3624      COM /History/ Status$[1],Time_orgn$[8],Date_orgn$[11]
3626      COM /History/ Time_chng$[8],Date_chng$[11],Description$[160]
3628    !
3630      COM /Labels/ Labels$(30)[60],INTEGER Lbl_count,REAL Lbl_addr(30,6)
3632    !Lbl_addr: x, y, pen, size, LDIR, LORG
3634    !
3636      COM /Data_param/ INTEGER Datacount,Filesize,Curvecount,Roster(17,4)
3638      COM /Data_param/ REAL Sym_size,Symbol$(17)[2],Curve_id$(17)[40]
3640      COM /Data_param/ REAL Xmin_data,Xmax_data
3642      COM /Data_param/ REAL Ymin_data,Ymax_data
3644    !
3646    !Roster: Curve#, Start Addr in File(*), Datacount, and PEN
3648    !Symbol$(i)="" or "Y" => no symbol, connect pts
3650    !Symbol$(i)="*Y" =>  * symbol, connect pts
3652    !Symbol$(i)="*N" =>  * symbol, do not connect pts
3654    !
3656      COM /Background/ Graphtype$[12],Margins$(2)[10],Papersize$[1]
3658      COM /Background/ REAL Pen_speed,INTEGER Backgnd_pen,Auto_time
3660      COM /Background/ INTEGER Auto_file,REAL X_cross_y,Y_cross_x
3662      COM /Background/ Xgrid_tick$[4],INTEGER Xmajor,Xminor
3664      COM /Background/ Ygrid_tick$[4],INTEGER Ymajor,Yminor
3666      COM /Background/ REAL Xmin_graph,Xmax_graph,Ymin_graph,Ymax_graph
3668    !
3670      COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
3672      COM /Interrupts/ INTEGER Intr_prty
```

```
3674      COM /Enlarge_file/ INTEGER Overflow
3676      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
3678 !
3680      INTEGER R,Hold_size,Local_prty,Allocated,Fls_cnt
3682      DIM Ac$[5],Tempfile$[10],Mask$[10],Ftype$[5],Fls$(1)[10]
3684      REAL Dtime
3686      OFF KEY
3688      Local_prty=Intr_prty
3690      !
3692      !Select the disk drive where the data exists
3694      !
3696      IF Overflow<>0 THEN Overflow=0
3698      Hold_size=0
3700      Dtime=0.
3702      Allocated=0
3704 Selectdrive:     !
3706      IF Diskdrive$="NO DISK" THEN Diskdrive$=""
3708      IF LEN(Diskdrive$)>0 THEN GOTO Choosefilename
3710      GRAPHICS OFF
3712      OUTPUT 2 USING "#,K";"K"
3714      CALL Select_disk
3716      IF Diskdrive$="NO DISK" THEN GOTO Mistakelineset
3718 Choosefilename: !
3720      Tempfile$=Filename$
3722      IF LEN(Filename$)>0 THEN GOTO Bring_in_data
3724      Ac$="CAT"
3726      CALL Enterfilename(Ac$)
3728      IF LEN(Filename$)=0 OR POS(Filename$,"*")>1 THEN
3730         IF POS(Filename$,"*")>1 THEN ! set mask$
3732            Mask$=Filename$[1,POS(Filename$,"*")-1]
3734            Filename$=""
3736         ELSE
3738            Mask$=""! no preselection
3740         END IF
3742         Ftype$="BDAT "  ! examine BDAT files only
3744         Fls_cnt=1  ! select one file
3746         Intr_prty=Local_prty+1
3748         CALL File_menu(Mask$,Ftype$,Fls$(*),Fls_cnt,0,0)
3750         Intr_prty=Local_prty
3752         Filename$=Fls$(1)
3754         IF LEN(Filename$)=0 THEN  ! aborted
3756            Filename$=Tempfile$
3758            GOTO Mistakelineset
3760         END IF
3762      END IF
3764 Bring_in_data: !
3766      !
3768      !Find this file on the disk.
3770      !
3772      ON ERROR GOTO Cant_findfile
3774      ASSIGN @Datapath TO Filename$&Diskdrive$
3776      OFF ERROR
3778      Dtime=TIMEDATE
3780      DISP " LOADING disk file: ";Filename$;" ... ";
3782      ON ERROR GOTO Bad_file
3784      ENTER @Datapath;Status$
```

```
3786      OFF ERROR
3788      ON ERROR GOTO Cant_findfile
3790      SELECT Status$
3792      CASE "Y"    ! All graphics/data parameters exist.REN 100,2
3794         DISP " Complete graph. "
3796         ENTER @Datapath;Time_orgn$,Date_orgn$
3798         ENTER @Datapath;Time_chng$,Date_chng$
3800         ENTER @Datapath;Description$
3802         ENTER @Datapath;Labels$(*),Lbl_count,Lbl_addr(*)
3804         ENTER @Datapath;Curve_id$(*),Symbol$(*)
3806         ENTER @Datapath;Roster(*),Curvecount
3808         ENTER @Datapath;Graphtype$,Margins$(*)
3810         ENTER @Datapath;X_cross_y,Y_cross_x
3812         ENTER @Datapath;Xgrid_tick$,Xmajor,Xminor
3814         ENTER @Datapath;Ygrid_tick$,Ymajor,Yminor
3816         ENTER @Datapath;Xmin_graph,Xmax_graph
3818         ENTER @Datapath;Ymin_graph,Ymax_graph
3820      CASE "N"     ! Only data parameters exist.
3822         DISP " RAW data. "
3824      CASE ELSE
3826 Bad_file: DISP CHR$(12)
3828         DISP "Data file is not recognized, entry aborted.";
3830         DISP " ...continue."
3832         BEEP
3834         PAUSE
3836         OFF ERROR
3838         GOTO Mistakelineset
3840      END SELECT
3842      !
3844      ENTER @Datapath;Data_id$
3846      ENTER @Datapath;Datacount
3848      ENTER @Datapath;Hold_size
3850      IF NOT Allocated THEN
3852         IF Datacount>=1 AND Hold_size>=1 THEN
3854            ALLOCATE Holding_file(Hold_size,2)
3856         ELSE
3858            ALLOCATE Holding_file(1,2)
3860         END IF
3862         Allocated=1
3864      END IF
3866      ENTER @Datapath;Holding_file(*)
3868      ASSIGN @Datapath TO *
3870      OFF ERROR
3872      IF Datacount=0 THEN Mistakeline
3874      !
3876      !Copy data from Holding_file(*) to Basket_file(*)
3878      !
3880      MAT Basket_file= (0.)
3882      IF Datacount>Basketsize THEN   !Receiving file too small.
3884         Allocated=0
3886         DEALLOCATE Holding_file(*)
3888         DISP " DATA FILE overflow, new data discarded. ";
3890         DISP " (continue) "
3892         BEEP
3894         PAUSE
3896         IF Status$="Y" THEN
```

```
3898            Curvecount=0
3900            MAT Roster= (0)
3902         END IF
3904         Overflow=Hold_size
3906         GOTO Mistakelineset
3908      END IF
3910 Copydatafile:      !
3912      FOR R=1 TO Datacount
3914         Basket_file(R,1)=Holding_file(R,1)
3916         Basket_file(R,2)=Holding_file(R,2)
3918      NEXT R
3920      Basketsize=Datacount
3922      GOTO Mistakeline
3924      !
3926 Mistakelineset:Basketsize=0
3928 Mistakeline:OFF KEY
3930      IF Allocated THEN DEALLOCATE Holding_file(*)
3932      LOOP
3934      EXIT IF TIMEDATE-Dtime>1.8
3936      END LOOP
3938      DISP CHR$(12)
3940      OUTPUT 2 USING "#,K";"K"
3942      SUBEXIT
3944      !
3946      ! ////////////////////////////////////////////////////
3948      !
3950 Cant_findfile:      !Error in searching for the file.
3952      BEEP 500,.6
3954      SELECT ERRN
3956      CASE 56
3958         DISP "That file does not exist on this disk  ";
3960      CASE 72,73,76,82
3962         DISP Diskdrive$;" has failed or is not available ";
3964      CASE ELSE
3966         DISP ERRM$;
3968      END SELECT
3970      DISP " ....CONTINUE to try again."
3972      PAUSE
3974      Filename$=""
3976      Diskdrive$=""
3978      GOTO Selectdrive
3980      !
3982   SUBEND
3984   !
3986   ! ****************************************************************
3988   !
```

```
100   ! RE-STORE "GAUSS:,1400"
102   !
104    COM /Sys/ Sys_id$[10]
106    COM /Sys_msi/ Msi_id$[20]
108   !
110    OUTPUT KBD USING "K,#";"SCRATCH KEYE"        !ERASE SOFT KEYS
112    CONTROL KBD,15;0! sets the color of the soft keys
114    CONTROL KBD,2;1
116   !
118    !ntr_prty=1
120    CLEAR SCREEN
122    CALL Gauss
124   !
126    OUTPUT KBD USING "K,#";"LOAD KEYE"! restore the typing aid keys
128    PRINT TABXY(1,5);"END of program. So long."
130    MASS STORAGE IS ":,1400"
132   !
134    END
136   !
138   !_____________________________________________________________
140   !
142    SUB Gauss
144   !
146 Gauss:  !
148       OPTION BASE 1
150       COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
152       COM /Interrupts/ INTEGER Intr_prty
154       !
156       ! Declare variables
158       INTEGER Fs,Ptn,Filesize,Datacount,Num
160       REAL Sigma,Pulse_amp,Time_window,Wv(32767,2),Time_var,Temp
162       DIM Data_id$[40]
164       !
166       ! Give information and ask for a value of sigma
168       PRINT "This program produces an impulse-like, Gaussian waveform."
170       PRINT "You will need to input a value for sigma."
172        !
174       GOSUB Sigma_in
176       !
178       GOSUB Num_pnts
180       REDIM Wv(Num,2)
182       !
184       ! We want to create a gaussian, so starting from the equation
186       ! for a gaussian exp(-0.5 (x/sigma)^^2 ) we find the value of the
188       ! FWHM.  At this point 0.5 = exp(-0.5 ((x-mean)/sigma)^^2) so
190       ! [HALF MAX] = 0.693147181 = (-0.5) * ((x - mean)/sigma)^^2 and
192       ! (x - mean) = 2 * 1.17741 * sigma.
194       !
196       PRINT "For this sigma, full width-half max = ",2.35482*Sigma
198       !
200       ! Compute the pulse amplitude
202       !
204       GOSUB Pls_amp
```

```
206       !
208       ! Ask for the time window in seconds and ask for the filename
210       !
212       GOSUB Get_wndow
214       !
216       !Calculate the standard normalizing factor for gaussian
218       !
220       Dt=Time_window/Num
222       Time_var=0.
224       Two_sig_squared=2.0*Sigma*Sigma
226       IF (10.729*Sigma/Dt+.5)>32767 THEN
228          DISP "The calculation cannot be completed as requested. Please"
230          WAIT 1.0
232          DISP "check your numbers and run the program again."
234          SUBEXIT
236       ELSE
238          Ptn=INT(10.729*Sigma/Dt+.5)+1
240       !Calculate the points in the gaussian
242       END IF
244       !
246       FOR I=1 TO Num
248          Exponent=((I-Ptn)*(I-Ptn)*Dt*Dt/Two_sig_squared)
250          IF Exponent>57.56 THEN
252             Wv(I,2)=0.
254          ELSE
256             Wv(I,2)=Pulse_amp*EXP(-Exponent)
258          END IF
260          Wv(I,1)=Time_var
262          Time_var=Time_var+Dt
264       NEXT I
266       !
268       Filesize=Num
270       Datacount=Num
272       Filename$=""
274       Diskdrive$=""
276       Data_id$="Gaussian waveform"
278       !
280       CALL Data_to_disk_r(Wv(*),Filesize,Datacount,Data_id$)
282       !
284       !Exit the subroutine
286       !
288       CLEAR SCREEN
290       SUBEXIT
292 Sigma_err: !
294       BEEP
296       DISP "ERROR IN VALUE OF SIGMA. TRY AGAIN."
298       WAIT 1.0
300 Sigma_in:  !
302       ON ERROR GOSUB Sigma_err
304       Test$=""
306       INPUT "Please enter a value for sigma: ",Test$
308       IF LEN(Test$)<1 THEN GOTO Sigma_err
310       CALL Data_check(Test$)
312       Sigma=VAL(Test$)
314       OFF ERROR
316       IF Sigma<=0 THEN GOTO Sigma_err
```

```
318       RETURN
320 Num_err: !
322       BEEP
324       DISP "ERROR IN THE NUMBER OF POINTS, TRY AGAIN"
326       WAIT 1.0
328 Num_pnts: !
330       Test$ = ""
332       ON ERROR GOTO Num_err
334       INPUT "Enter an integer number for the points in the waveform?",Test$
336       IF LEN(Test$) < 1 THEN GOTO Num_err
338       Temp = VAL(Test$)
340       OFF ERROR
342       IF (Temp < = 0) OR (INT(Temp) < > Temp) THEN GOTO Num_err
344       Num = INT(Temp)
346       RETURN
348 Ampl_err:      !
350       BEEP
352       DISP "INPUT ERROR, PLEASE TRY AGAIN."
354       WAIT 1.0
356 Pls_amp:      !
358       ON ERROR GOTO Ampl_err
360 Inp:  Test$ = ""
362       INPUT "Would you like a unit area pulse? y/n (default is y).",Test$
364       IF LEN(Test$) < 1 THEN Test$ = "y"
366       !
368       IF (Test$ < > "y") AND (Test$ < > "Y") AND (Test$ < > "N") AND (Test$ < > "n") THEN
370          BEEP
372          DISP "I don't understand, please try again. This time answer y/n."
374          WAIT 1.0
376          GOTO Inp
378       END IF
380       OFF ERROR
382       IF (Test$ = "Y") OR (Test$ = "y") THEN
384          Pulse_amp = 1.0/(SQRT(2.0*PI)*Sigma)
386       ELSE
388          GOSUB Input_amp
390       END IF
392       RETURN
394 Pnt_err:      !
396       BEEP
398       DISP "INPUT ERROR, PLEASE TRY AGAIN."
400       WAIT 1.0
402 Point_place:  !
404       ON ERROR GOTO Pnt_err
406       INPUT "At what point would you like the maximum to occur? ",Test$
408       IF LEN(Test$) < 1 THEN GOTO Pnt_err
410       Temp = VAL(Test$)
412       OFF ERROR
414       IF (Temp < = 0) OR (INT(Temp) < > Temp) THEN GOTO Pnt_err
416       Ptn = INT(Temp)
418       RETURN
420 Window_err:      !
422       BEEP
424       DISP "TIME WINDOW INPUT ERROR, PLEASE TRY AGAIN."
426       WAIT 1.0
428 Get_wndow:      !
```

```
430       ON ERROR GOTO Window_err
432       Test$=""
434       INPUT "What is the time window in seconds? ",Test$
436       IF LEN(Test$)<1 THEN GOTO Window_err
438       CALL Data_check(Test$)
440       Time_window=VAL(Test$)
442       IF Time_window<=0 THEN
444          DISP "Time window must be greater than zero."
446          WAIT 1.0
448          GOTO Window_err
450       END IF
452       OFF ERROR
454       RETURN
456 Inp_amp_err:      !
458       BEEP
460       DISP "INPUT ERROR, PLEASE TRY AGAIN."
462       WAIT 1.0
464 Input_amp:      !
466       ON ERROR GOTO Inp_amp_err
468       Test$=""
470       INPUT "Enter in the desired pulse amplitude.",Test$
472       IF LEN(Test$)<1 THEN GOTO Inp_amp_error
474       CALL Data_check(Test$)
476       Pulse_amp=VAL(Test$)
478       IF Pulse_amp=0 THEN GOTO Inp_amp_err
480       OFF ERROR
482       RETURN
484    SUBEND
486    !
488    !!!!!!!!!!!!!!!
490    SUB Data_to_disk_r(REAL File(*),INTEGER Filesize,Datacount,Data_id$)
492 Data_to_disk_r:   ! Original: 13 Nov 1984
494                   ! Revision: 06 Aug 1987
496       ! This routine will SAVE data files on the disk in RAW data format.
498       ! Special features:
500       !    If the Diskdrive$ and/or the Filename$ are null this routine
502       !    will prompt the operator for information.  However, if they
504       !    are not null it is assumed that the program is supplying the
506       !    correct information.
508       !
510       OPTION BASE 1
512       COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
514       COM /Interrupts/ INTEGER Intr_prty
516       INTEGER Local_prty,Diskspace
518       DIM Ac$[5],Status$[1]
520       REAL Dtime
522       OFF KEY
524       Local_prty=Intr_prty
526       Dtime=0.
528       !
530       !Select the disk drive for data storage
532       !
534 Selectdrive:  !
536       IF Diskdrive$="NO DISK" THEN Diskdrive$=""
538       IF LEN(Diskdrive$)>0 THEN GOTO Choosefilename
540       GRAPHICS OFF
```

```
542       OUTPUT 2 USING "#,K";"K"
544       CALL Select_disk
546       IF Diskdrive$="NO DISK" THEN GOTO Mistakeline
548 Choosefilename:  !
550       IF LEN(Filename$)>0 THEN GOTO Send_to_disk
552       Ac$="ABORT"
554       CALL Enterfilename(Ac$)
556       IF LEN(Filename$)=0 THEN GOTO Mistakeline
558 Send_to_disk:  ! Create file and save information.
560       ON ERROR GOTO Cant_savedata
562       Diskspace=INT((Filesize*16.0)/256)+2
564       CREATE BDAT Filename$&Diskdrive$,Diskspace,256
566       Dtime=TIMEDATE
568       DISP " SAVING data in file ";Filename$;" on ";Diskdrive$
570       Status$="N"
572       ASSIGN @Datapath TO Filename$&Diskdrive$
574       OUTPUT @Datapath;Status$
576       OUTPUT @Datapath;Data_id$    !40 chrs description of data
578       OUTPUT @Datapath;Datacount   !number of xy points
580       OUTPUT @Datapath;Filesize    !size of array
582       OUTPUT @Datapath;File(*)
584       ASSIGN @Datapath TO *
586       OFF ERROR
588       !
590 Mistakeline:OFF KEY
592       LOOP
594       EXIT IF TIMEDATE-Dtime>1.8
596       END LOOP
598       DISP CHR$(12)
600       OUTPUT 2 USING "#,K";"K"
602       SUBEXIT
604       !
606       ! ////////////////////////////////////////////////////
608       !
610 Cant_savedata:  !
612       BEEP 500,.6
614       SELECT ERRN
616       CASE 72,73,76,78,81,82,90,93
618          DISP Diskdrive$;" has failed or is not available ";
620          DISP " ....CONTINUE to try again."
622          PAUSE
624          Diskdrive$=""
626       CASE 84,85
628          DISP " This disk is not initialized ";
630          DISP " ....CONTINUE to try again."
632          PAUSE
634          Diskdrive$=""
636       CASE 55,64
638          DISP " This disk is full, insert new floppy and/or";
640          DISP " select new drive ...CONTINUE "
642          PAUSE
644          Diskdrive$=""
646       CASE ELSE
648          CALL Errortrap
650          IF LEN(Filename$)>0 THEN GOTO Send_to_disk
652       END SELECT
```

```
654     GOTO Selectdrive
656     !
658   SUBEND
660   !
662   ! ************************************************************
664   !
666   SUB Errortrap
668 Errortrap:  ! Original: 13 Nov 1984
670           ! Revision: 06 Aug 1987
672           ! Trap most disk errors here
674     COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
676     DIM File$[20],Test$[160],What$[20],Ac$[5]
678     BEEP 400,.6
680     SELECT ERRN
682     CASE 54
684       DISP "DUPLICATE FILE NAME:  ";Filename$;
686       DISP "....PURGE old one? (Y/N)";
688       LINPUT What$
690       What$=TRIM$(What$)
692       SELECT What$[1,1]
694       CASE "Y","y"
696         PURGE Filename$&Diskdrive$
698       CASE ELSE
700         Ac$="VALID"
702         CALL Enterfilename(Ac$)
704       END SELECT
706     CASE 52,53
708       DISP "Improper FILE NAME -- ENTER NEW FILE NAME";
710       OUTPUT 2 USING "#,K,K";"#";Filename$
712       LINPUT Filename$
714       Filename$=TRIM$(Filename$)
716     CASE 56
718       DISP "FILE:  ";Filename$;" is not on this disk, please insert";
720       DISP " correct disk"
722       PAUSE
724     CASE 64
726       DISP "This disk is full, PLEASE insert clean disk"
728       PAUSE
730     CASE 56
732       DISP "DATA INPUT disk must be in drive!!  ";
734       DISP "...CONTINUE when ready."
736       PAUSE
738     CASE 72,73,76
740       DISP Diskdrive$;
742       DISP " is not available, type correct";
744       DISP " unit specifier (ie. ':,707,0')."
746       OUTPUT 2 USING "K,#";Diskdrive$
748       LINPUT Diskdrive$
750     CASE 80
752       DISP "CHECK DISK drive door!"
754       PAUSE
756     CASE ELSE
758       DISP ERRM$;" 'CONTINUE' when fixed"
760       PAUSE
762     END SELECT
764     DISP CHR$(12)
```

```
766       SUBEXIT
768     SUBEND
770     !
772     ! ************************************************************
774     !
776     SUB Menu_scroll(D$,T$,Items$(*),INTEGER Item_cnt,To_select,Choose(*))
778   Menu_scroll:! Original: 22 Jun 1987, Galen Koepke, NBS 723.04
780             ! Revision: 06 Aug 1987, 10:00
782             !
784             ! A general purpose menu utility for scrolling items and
786             ! selecting a given number of them.
788             ! The items are arranged in screens of 15 items each and
790             ! the user may access screens via softkeys. There may be
792             ! up to 10 screens or 150 items to choose from.
794             ! Items$(*) contains the item descriptions
796             ! Item_cnt is the number of items in Items$(*)
798             ! Choose(*) is dimensioned to the number of required choices
800             !         and will be filled with the item numbers chosen.
802             ! To_select is the number of required choices.
804             !
806       OPTION BASE 1
808       PRINTER IS CRT
810       DEG
812       GOSUB Def_variables
814       GOSUB Define_screens
816       GOSUB Make_selections
818       IF Null_file THEN  ! reset to zero
820          Item_cnt=0
822          Items$(1)=""
824          To_select=0    ! no valid selections
826       END IF
828       SUBEXIT
830       !
832       ! ////////////////////////////////////////////////
834       !
836   Def_variables:!
838       COM /Interrupts/ INTEGER Intr_prty
840       COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
842       COM /Sys/ Sys_id$[10]
844       !
846       INTEGER Screen_cnt,Items_per_scn,First_item(10),Last_item(10)
848       INTEGER I,J,K,First_line,Last_line,Active_screen,Pointer,Last_pt
850       INTEGER Local_prty,Skips,Knobcount,Pointeractive,K0,Null_file
852       INTEGER Exit_flag
854       DIM Marker$[8],Test$[160]
856       !
858       ! initialize parameters
860       !
862       Local_prty=Intr_prty
864       IF Local_prty<1 THEN Local_prty=10
866       IF LEN(Sys_id$)=0 THEN Sys_id$=SYSTEM$("SYSTEM ID")
868       IF Item_cnt<1 THEN
870          Null_file=1
872          Item_cnt=1
874          To_select=0
876          Items$(1)="**** Empty ****"
```

```
878       ELSE
880          Null_file=0
882       END IF
884       IF To_select>Item_cnt THEN To_select=Item_cnt
886       Skips=0
888       Knobcount=0
890       Doneflag=0
892       Marker$="===>"&RPT$(CHR$(8),4)
894       RETURN
896       !
898       ! ////////////////////////////////////////////////
900       !
902 Define_screens:! Set up screens of 15 items each.
904             !
906       Items_per_scn=15    ! Maximum number of displayable items
908       IF INT(Item_cnt/Items_per_scn)=Item_cnt/Items_per_scn  THEN
910          Screen_cnt=INT(Item_cnt/Items_per_scn)
912       ELSE
914          Screen_cnt=INT(Item_cnt/Items_per_scn)+1
916       END IF
918       J=1
920       FOR I=1 TO Screen_cnt   ! set up each screen
922          First_item(I)=J
924          IF J+Items_per_scn-1<Item_cnt THEN
926             Last_item(I)=J+Items_per_scn-1
928             J=J+Items_per_scn
930          ELSE
932             Last_item(I)=Item_cnt
934          END IF
936       NEXT I
938       RETURN
940       !
942       ! ////////////////////////////////////////////////
944       !
946 Make_selections:! MENU setup and use.
948       Active_screen=1   ! first screen is active
950       First_line=2      ! first printed line on screen = 2 or greater.
952       GOSUB Write_screen  ! activate screen at Active_screen
954                           ! and set First_line and Last_line for Pointer
956                           ! write Marker$ to first non-selected line.
958       K0=0                ! Keys start at zero
960       Exit_flag=0         ! allow ENTER key to exit when selections filled.
962       IF Sys_id$[1,4]="S300" THEN
964          CONTROL KBD,2;1
966          STATUS KBD,14;J
968          IF J=0 THEN       ! key 1 defined
970             K0=1
972          ELSE              ! key 0 defined
974             K0=0
976          END IF
978       ELSE
980          K0=0
982       END IF
984 Key_loop:   !
986       ON KBD,Local_prty GOSUB Process_kbd
988       ON KNOB .01,Local_prty GOSUB Move_pointer
```

```
990       IF Skips<To_select THEN
992          DISP D$
994          IF To_select>1 THEN
996             Test$=" Select  "&VAL$(Skips+1)&" of "&VAL$(To_select)
998          ELSE
1000            Test$=" Select"
1002         END IF
1004         ON KEY K0 LABEL Test$,Local_prty GOSUB Select_item
1006      ELSE
1008         IF To_select>0 THEN
1010            DISP " Selection process complete ..."
1012         ELSE
1014            DISP " Menu for information only ... "
1016         END IF
1018         ON KEY K0 LABEL "Accept",Local_prty GOTO Exit_line
1020      END IF
1022      IF Active_screen<Screen_cnt THEN
1024         ON KEY K0+1 LABEL " Next  Screen",Local_prty GOSUB Next_screen
1026      ELSE
1028         OFF KEY K0+1
1030      END IF
1032      IF Active_screen>1 THEN
1034         ON KEY K0+2 LABEL " Last  Screen",Local_prty GOSUB Last_screen
1036      ELSE
1038         OFF KEY K0+2
1040      END IF
1042      IF Skips>0 THEN
1044         ON KEY K0+3 LABEL " Reset  Select",Local_prty GOSUB Select_reset
1046      ELSE
1048         OFF KEY K0+3
1050      END IF
1052      IF To_select>0 THEN
1054         ON KEY K0+4 LABEL " Abort ",Local_prty GOTO Escape_line
1056      ELSE
1058         OFF KEY K0+4
1060      END IF
1062      IF Exit_flag THEN Exit_line
1064      GOTO Key_loop
1066 Escape_line:Skips=0
1068      MAT Choose= (0)
1070      To_select=0
1072 Exit_line:OFF KEY
1074      OFF KNOB
1076      OFF KBD
1078      OUTPUT KBD;CHR$(255)&CHR$(75);
1080      PRINT CHR$(128);
1082      ! everything cleared, now go back to work.
1084      RETURN
1086      !
1088      ! ////////////////////////////////////////////////
1090      !
1092 Next_screen:     !
1094      OFF KBD
1096      OFF KNOB
1098      OFF KEY
1100      IF Active_screen=Screen_cnt THEN RETURN
```

```
1102      Active_screen=Active_screen+1
1104      GOSUB Write_screen
1106      RETURN
1108      !
1110      ! ////////////////////////////////////////////////
1112      !
1114 Last_screen:      !
1116      OFF KBD
1118      OFF KNOB
1120      OFF KEY
1122      IF Active_screen=1 THEN RETURN
1124      Active_screen=Active_screen-1
1126      GOSUB Write_screen
1128      RETURN
1130      !
1132      ! ////////////////////////////////////////////////
1134      !
1136 Select_item:!
1138      OFF KBD
1140      OFF KNOB
1142      OFF KEY
1144      IF NOT Pointeractive THEN
1146         DISP "NO additional selections for this screen."
1148         BEEP
1150         WAIT 2
1152         DISP CHR$(12);
1154         RETURN
1156      END IF
1158      IF Skips=To_select THEN
1160         IF To_select=0 THEN
1162            DISP "This menu is for information only,";
1164            DISP " no selection allowed."
1166         ELSE
1168            DISP "All selections have been filled,";
1170            DISP " 'Select Reset' to repeat."
1172         END IF
1174         BEEP
1176         WAIT 2
1178         DISP CHR$(12);
1180         RETURN
1182      END IF
1184      Skips=Skips+1
1186      Choose(Skips)=First_item(Active_screen)+Pointer-First_line
1188      PRINT CHR$(129);  ! inverse video
1190      PRINT TABXY(10,Pointer);Items$(Choose(Skips))
1192      PRINT CHR$(128);
1194      PRINT TABXY(1,Pointer);
1196      SELECT Pointer
1198      CASE First_line
1200         GOSUB Point_forward
1202      CASE Last_line
1204         GOSUB Point_backward
1206      CASE ELSE
1208         ! move forward unless it requires wrapping to beginning.
1210         IF Skips-1>0 THEN  ! check for selected items.
1212            I=Pointer-First_line
```

```
1214            LOOP
1216               K=0
1218               FOR J=1 TO Skips
1220                  IF First_item(Active_screen)+I=Choose(J) THEN K=1
1222               NEXT J
1224            EXIT IF K=0
1226               I=I+1
1228               IF I+First_line>Last_line THEN K=-1
1230            EXIT IF K=-1
1232            END LOOP
1234            IF K=0 THEN
1236               GOSUB Point_forward
1238            ELSE
1240               GOSUB Point_backward
1242            END IF
1244         ELSE
1246            GOSUB Point_forward
1248         END IF
1250      END SELECT
1252      RETURN
1254      !
1256      ! ////////////////////////////////////////////////
1258      !
1260 Select_reset:      !Clear Choose file
1262      OFF KBD
1264      OFF KNOB
1266      OFF KEY
1268      Skips=0
1270      MAT Choose= (0)
1272      GOSUB Write_screen
1274      RETURN
1276      !
1278      ! ////////////////////////////////////////////////
1280      !
1282 Process_kbd:! Allow use of arrows and enter key in addition to soft.
1284      Test$=KBD$
1286      IF LEN(Test$)=1 AND Test$[1,1]<>CHR$(32) THEN
1288         BEEP 80,,.1
1290         RETURN
1292      END IF
1294      IF Test$[1,1]=CHR$(32) THEN GOSUB Point_forward
1296      IF Test$[1,1]<>CHR$(255) THEN RETURN
1298      SELECT Test$[2,2]
1300      CASE CHR$(255)
1302            ! do nothing
1304      CASE "V","T"
1306         GOSUB Point_forward
1308      CASE "^","W"
1310         GOSUB Point_backward
1312      CASE "E"
1314         IF Skips<To_select THEN
1316            GOSUB Select_item
1318         ELSE
1320            ! exit routine
1322            Exit_flag=1
1324         END IF
```

```
1326      CASE ELSE
1328         BEEP 80.,.1
1330      END SELECT
1332      Test$=""
1334      RETURN
1336      !
1338      ! ////////////////////////////////////////////////
1340      !
1342 Point_forward:Knobcount=5
1344      GOSUB Move_pointer
1346      RETURN
1348 Point_backward:Knobcount=-5
1350      GOSUB Move_pointer
1352      RETURN
1354      !
1356      ! ////////////////////////////////////////////////
1358      !
1360 Jog_pointer:! Move the selection pointer on the active screen.
1362           ! without regard to selected values
1364      IF Knobcount>0 THEN  ! Move forward
1366         Pointer=Pointer+1
1368      ELSE                 ! Move backward
1370         Pointer=Pointer-1
1372      END IF
1374      IF Pointer<First_line THEN Pointer=Last_line
1376      IF Pointer>Last_line THEN Pointer=First_line
1378      RETURN
1380      !
1382      ! ////////////////////////////////////////////////////
1384      !
1386 Move_pointer:! Control pointer to avoid re-selection of items
1388      IF NOT Pointeractive THEN RETURN  ! No selections to be made.
1390      Knobcount=Knobcount+KNOBX+KNOBY
1392      IF ABS(Knobcount)<4 THEN RETURN
1394      Last_pt=Pointer
1396      GOSUB Jog_pointer
1398      IF Skips>0 THEN
1400         LOOP
1402            J=Pointer-First_line
1404            FOR I=1 TO Skips
1406               IF First_item(Active_screen)+J=Choose(I) THEN J=999
1408            NEXT I
1410            IF J=999 AND Pointer=Last_pt THEN Pointeractive=0
1412         EXIT IF Pointeractive=0
1414            IF J=999 THEN GOSUB Jog_pointer
1416         EXIT IF J<>999
1418         END LOOP
1420      END IF
1422      Knobcount=0
1424      OUTPUT KBD;CHR$(255)&CHR$(84);  ! Bring screen home
1426      IF Last_pt=Last_line THEN PRINT CHR$(132);
1428      PRINT "    ";
1430      IF Pointeractive THEN ! Pointer active
1432         IF Pointer=Last_line THEN
1434            PRINT CHR$(132);
1436         ELSE
```

```
1438            PRINT CHR$(128);
1440          END IF
1442          PRINT TABXY(1,Pointer);Marker$;CHR$(128);
1444       END IF
1446       RETURN
1448       !
1450       ! ////////////////////////////////////////////////
1452       !
1454 Write_screen:! Write the screen pointed to by Active_screen
1456       ! home and clear screen
1458       OUTPUT KBD;CHR$(255)&CHR$(84)&CHR$(255)&CHR$(75);
1460       Knobcount=KNOBX+KNOBY   ! Clear knob and keyboard
1462       Knobcount=0
1464       Test$=KBD$
1466       Test$=""
1468       !
1470       PRINT TABXY(1,First_line-1);CHR$(132);" Item #| Screen #";
1472       PRINT USING "#,2D,3A,2D,3A";Active_screen," of ";Screen_cnt;" | "
1474       PRINT T$;RPT$(" ",52-LEN(T$));CHR$(128);
1476       J=0
1478       REPEAT
1480          IF J=Last_item(Active_screen)-First_item(Active_screen) THEN
1482             PRINT CHR$(132);
1484             PRINT TABXY(1,First_line+J);RPT$(" ",80)
1486          ELSE
1488             PRINT CHR$(128);
1490          END IF
1492          PRINT TABXY(5,First_line+J);
1494          PRINT USING "3D,A,#";First_item(Active_screen)+J,"|"
1496          IF Skips>0 THEN  ! make this line inverse video
1498             FOR I=1 TO Skips
1500                IF First_item(Active_screen)+J=Choose(I) THEN
1502                   PRINT CHR$(129);
1504                END IF
1506             NEXT I
1508          END IF
1510          PRINT TABXY(10,First_line+J);Items$(First_item(Active_screen)+J)
1512          J=J+1
1514       UNTIL J>=(Last_item(Active_screen)-First_item(Active_screen)+1)
1516       Last_line=Last_item(Active_screen)-First_item(Active_screen)
1518       Last_line=Last_line+First_line
1520       !
1522       ! set marker to first non-selected item.
1524       !
1526       Pointeractive=0
1528       IF To_select>0 THEN Pointeractive=1
1530       IF Skips>0 AND Pointeractive=1 THEN ! find first non-selected item
1532          J=0
1534          LOOP
1536             Pointer=First_line+J
1538             FOR I=1 TO Skips
1540                IF First_item(Active_screen)+J=Choose(I) THEN Pointer=0
1542             NEXT I
1544          EXIT IF Pointer<>0
1546             J=J+1
1548             IF First_line+J>Last_line THEN
```

```
1550            Pointeractive=0
1552            Pointer=First_line
1554          END IF
1556        EXIT IF Pointer<>0
1558        END LOOP
1560      ELSE
1562        Pointer=First_line
1564      END IF
1566      IF Pointeractive THEN
1568        IF Pointer=Last_line THEN
1570          PRINT CHR$(132);
1572        ELSE
1574          PRINT CHR$(128);
1576        END IF
1578        PRINT TABXY(1,Pointer);Marker$;CHR$(128);
1580      END IF
1582      RETURN
1584    SUBEND
1586    !
1588    ! ************************************************************
1590    !
1592    SUB File_menu(Mask$,Ftype$,Fls$(*),INTEGER Fls_cnt,Dir_on,Prt_on)
1594  File_menu:  !
1596      ! Original: 29 Jun 1987, G. Koepke
1598      ! Revision: 06 Aug 1987, 10:00
1600      OPTION BASE 1
1602      DEG
1604      COM /Sys/ Sys_id$[10]
1606      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
1608      COM /Interrupts/ INTEGER Intr_prty
1610      DIM Directory$(150)[80],Bd$(150)[71]
1612      DIM D$[80],T$[52],Ids$[40],Stat$[1]
1614      INTEGER Bd_cnt,File_cnt,I,C_cnt,C0(1),Format_error
1616      IF Fls_cnt>0 THEN ALLOCATE INTEGER Choose(Fls_cnt)
1618      !
1620      ! Catalog the disk specified
1622      !
1624      ON ERROR GOTO Cat_errors
1626      DISP " Reading the Directory ... "
1628      MASS STORAGE IS Diskdrive$
1630      CAT TO Directory$(*);NO HEADER,COUNT File_cnt
1632      OFF ERROR
1634      !
1636      ! set up array of legal file name$.
1638      !
1640      Bd_cnt=0
1642      FOR I=1 TO File_cnt
1644        IF Directory$(I)[32,36]=Ftype$ THEN  ! Ftype$="BDAT "
1646                                             ! Ftype$="PROG "
1648          IF LEN(Mask$)>0 THEN ! Test for mask$
1650            IF Directory$(I)[1,LEN(Mask$)]=Mask$ THEN
1652              Bd_cnt=Bd_cnt+1
1654              Bd$(Bd_cnt)=Directory$(I)[1;10]
1656            END IF
1658          ELSE
1660            Bd_cnt=Bd_cnt+1
```

```
1662              Bd$(Bd_cnt)=Directory$(I)[1;10]
1664           END IF
1666        END IF
1668     NEXT I
1670     !
1672     ! set up file menu
1674     !
1676     D$="Select "&VAL$(Fls_cnt)&" file names for data entry."
1678     T$="List of "&Ftype$&"files on "&Diskdrive$
1680     IF LEN(Mask$)>0 THEN
1682        T$=T$&" mask="&Mask$
1684     END IF
1686     IF Bd_cnt>0 THEN
1688        IF Dir_on>0 THEN GOSUB Read_data_id
1690        IF Prt_on THEN
1692           GOSUB List_directory
1694        ELSE
1696           C_cnt=Fls_cnt
1698           DISP CHR$(12)
1700           IF Fls_cnt>0 THEN
1702              CALL Menu_scroll(D$,T$,Bd$(*),Bd_cnt,C_cnt,Choose(*))
1704           ELSE
1706              CALL Menu_scroll(D$,T$,Bd$(*),Bd_cnt,C_cnt,C0(*))
1708           END IF
1710           !
1712           ! transfer file names to Fls$(*).
1714           !
1716           IF C_cnt=0 THEN ! selection process aborted
1718              MAT Fls$= ("")
1720           ELSE
1722              MAT SORT Choose(*)
1724              FOR I=1 TO C_cnt
1726                 Fls$(I)=Bd$(Choose(I))[1;10]
1728              NEXT I
1730           END IF
1732        END IF
1734     ELSE
1736        DISP " This directory contains no BDAT files ... "
1738        WAIT 2.5
1740     END IF
1742     DISP CHR$(12)
1744     SUBEXIT
1746 Cat_errors:!
1748     DISP "ERROR ... ";ERRM$
1750     BEEP
1752     PAUSE
1754     C_cnt=0
1756     MAT Fls$= ("")
1758     SUBEXIT
1760     !
1762     ! ////////////////////////////////////////////////////
1764     !
1766 Read_data_id: ! This routine expects to see Ids$ from
1768           ! GRAPH_DATA raw data files.
1770     DISP " Reading file contents ... "
1772     FOR I=1 TO Bd_cnt    ! each BDAT file
```

```
1774        Ids$="Data not recognized."
1776        ON ERROR GOTO Not_recognized
1778        ASSIGN @Io_path TO Bd$(I)[1;10]
1780        ENTER @Io_path;Stat$
1782        SELECT Stat$
1784        CASE "N"
1786           ENTER @Io_path;Ids$
1788        CASE "Y"
1790           Ids$="Complete graph ... use GRAPH_DATA."
1792        END SELECT
1794 Not_recognized:ASSIGN @Io_path TO *
1796        OFF ERROR
1798        IF Dir_on=2 THEN
1800           GOSUB Interpret_1
1802           IF Format_error THEN GOTO Other_format
1804           GOTO Go_on
1806        END IF
1808 Other_format:!
1810        Bd$(I)[11,71]=" ... "&Ids$
1812 Go_on:NEXT I
1814     RETURN
1816     !
1818     ! ////////////////////////////////////////////////
1820     !
1822 Interpret_1:! This is used to interpret TEM program ID strings.
1824     Format_error=0
1826     ! identify this particular format
1828     IF LEN(Ids$)<40 THEN
1830        Format_error=1
1832        RETURN
1834     END IF
1836     IF Ids$[40]<>"*" THEN
1838        Format_error=1
1840        RETURN
1842     END IF
1844     ! make the information readable
1846     Bd$(I)[11,15]=" ... "
1848     Bd$(I)[16,25]=Ids$[1,10]
1850     Bd$(I)[26,32]=", "&Ids$[11,12]&":"&Ids$[13,14]
1852     Bd$(I)[33,42]=", "&Ids$[15,16]&" "&Ids$[17,18]&" "&Ids$[19,20]
1854     Bd$(I)[43,55]=", "&Ids$[21,27]&" MHz"
1856     Bd$(I)[56,65]=", "&Ids$[28,33]&"vm"
1858     Bd$(I)[66,71]=","&Ids$[38,39]
1860     RETURN
1862     !
1864     ! ////////////////////////////////////////////////
1866     !
1868 List_directory: ! This routine will provide a tabular listing of
1870              ! the directory along with Ids$ if provided
1872              !
1874     DISP " Listing directory ... "
1876     PRINTER IS PRT
1878     PRINT USING "//"
1880     PRINT T$
1882     PRINT RPT$("-",80)
1884     PRINT "File name";
```

```
1886      IF Dir_on THEN
1888         PRINT "  ...  contents"
1890      ELSE
1892         PRINT
1894      END IF
1896      PRINT RPT$("-",80)
1898      FOR I=1 TO Bd_cnt
1900         PRINT Bd$(I)
1902      NEXT I
1904      PRINT RPT$("_",80)
1906      PRINT
1908      PRINTER IS CRT
1910      RETURN
1912   SUBEND
1914   !
1916   ! ***************************************************************
1918   !
1920   SUB Load_disk_data(Basket_file(*),INTEGER Basketsize,Data_id$,INTEGER Flg)
1922 Load_disk_data:  ! Original: 13 Nov 1984
1924                  ! Revision: 02 Dec 1987
1926      !This routine will enter data files from the disk
1928      OPTION BASE 1
1930      !
1932      COM /Sys/ Sys_id$
1934      COM /History/ Status$[1],Time_orgn$[8],Date_orgn$[11]
1936      COM /History/ Time_chng$[8],Date_chng$[11],Description$[160]
1938   !
1940      COM /Labels/ Labels$(30)[60],INTEGER Lbl_count,REAL Lbl_addr(30,6)
1942   !Lbl_addr: x, y, pen, size, LDIR, LORG
1944   !
1946      COM /Data_param/ INTEGER Datacount,Filesize,Curvecount,Roster(17,4)
1948      COM /Data_param/ REAL Sym_size,Symbol$(17)[2],Curve_id$(17)[40]
1950      COM /Data_param/ REAL Xmin_data,Xmax_data
1952      COM /Data_param/ REAL Ymin_data,Ymax_data
1954   !
1956   !Roster: Curve#, Start Addr in File(*), Datacount, and PEN
1958   !Symbol$(i)="" or "Y" => no symbol, connect pts
1960   !Symbol$(i)="*Y" =>  * symbol, connect pts
1962   !Symbol$(i)="*N" =>  * symbol, do not connect pts
1964   !
1966      COM /Background/ Graphtype$[12],Margins$(2)[10],Papersize$[1]
1968      COM /Background/ REAL Pen_speed,INTEGER Backgnd_pen,Auto_time
1970      COM /Background/ INTEGER Auto_file,REAL X_cross_y,Y_cross_x
1972      COM /Background/ Xgrid_tick$[4],INTEGER Xmajor,Xminor
1974      COM /Background/ Ygrid_tick$[4],INTEGER Ymajor,Yminor
1976      COM /Background/ REAL Xmin_graph,Xmax_graph,Ymin_graph,Ymax_graph
1978   !
1980      COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
1982      COM /Interrupts/ INTEGER Intr_prty
1984      COM /Enlarge_file/ INTEGER Overflow
1986      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
1988      COM /Data_stuff/ INTEGER Number,REAL Delta_x
1990   !
1992      INTEGER R,Hold_size,Local_prty,Allocated,Fls_cnt
1994      DIM Ac$[5],Tempfile$[10],Mask$[10],Ftype$[5],Fls$(1)[14]
1996      REAL Dtime
```

```
1998      OFF KEY
2000      Local_prty=Intr_prty
2002      !
2004      !Select the disk drive where the data exists
2006      !
2008      IF Overflow<>0 THEN Overflow=0
2010      Hold_size=0
2012      Dtime=0.
2014      Allocated=0
2016 Selectdrive:      !
2018      IF Diskdrive$="NO DISK" THEN Diskdrive$=""
2020      IF LEN(Diskdrive$)>0 THEN GOTO Choosefilename
2022      GRAPHICS OFF
2024      OUTPUT 2 USING "#,K";"K"
2026      CALL Select_disk
2028      IF Diskdrive$="NO DISK" THEN GOTO Mistakelineset
2030 Choosefilename: !
2032      Tempfile$=Filename$
2034      IF LEN(Filename$)>0 THEN GOTO Bring_in_data
2036      Ac$="CAT"
2038      CALL Enterfilename(Ac$)
2040      IF LEN(Filename$)=0 OR POS(Filename$,"*")>1 THEN
2042         IF POS(Filename$,"*")>1 THEN ! set mask$
2044            Mask$=Filename$[1,POS(Filename$,"*")-1]
2046            Filename$=""
2048         ELSE
2050            Mask$=""! no preselection
2052         END IF
2054         Ftype$="BDAT "  ! examine BDAT files only
2056         Fls_cnt=1  ! select one file
2058         Intr_prty=Local_prty+1
2060         CALL File_menu(Mask$,Ftype$,Fls$(*),Fls_cnt,0,0)
2062         Intr_prty=Local_prty
2064         Filename$=Fls$(1)
2066         IF LEN(Filename$)=0 THEN  ! aborted
2068            Filename$=Tempfile$
2070            GOTO Mistakelineset
2072         END IF
2074      END IF
2076 Bring_in_data: !
2078      !
2080      !Find this file on the disk.
2082      !
2084      ON ERROR GOTO Cant_findfile
2086      ASSIGN @Datapath TO Filename$&Diskdrive$
2088      OFF ERROR
2090      Dtime=TIMEDATE
2092      DISP " LOADING disk file: ";Filename$;" ... ";
2094      ON ERROR GOTO Bad_file
2096      ENTER @Datapath;Status$
2098      OFF ERROR
2100      ON ERROR GOTO Cant_findfile
2102      SELECT Status$
2104      CASE "Y"    ! All graphics/data parameters exist.REN 100,2
2106         DISP " Complete graph. "
2108         ENTER @Datapath;Time_orgn$,Date_orgn$
```

```
2110        ENTER @Datapath;Time_chng$,Date_chng$
2112        ENTER @Datapath;Description$
2114        ENTER @Datapath;Labels$(*),Lbl_count,Lbl_addr(*)
2116        ENTER @Datapath;Curve_id$(*),Symbol$(*)
2118        ENTER @Datapath;Roster(*),Curvecount
2120        ENTER @Datapath;Graphtype$,Margins$(*)
2122        ENTER @Datapath;X_cross_y,Y_cross_x
2124        ENTER @Datapath;Xgrid_tick$,Xmajor,Xminor
2126        ENTER @Datapath;Ygrid_tick$,Ymajor,Yminor
2128        ENTER @Datapath;Xmin_graph,Xmax_graph
2130        ENTER @Datapath;Ymin_graph,Ymax_graph
2132      CASE "N"     ! Only data parameters exist.
2134        DISP " RAW data. "
2136      CASE ELSE
2138 Bad_file: DISP CHR$(12)
2140        DISP "Data file is not recognized, entry aborted.";
2142        DISP " ...continue."
2144        BEEP
2146        PAUSE
2148        OFF ERROR
2150        GOTO Mistakelineset
2152      END SELECT
2154      !
2156      ENTER @Datapath;Data_id$
2158      IF Flg THEN
2160        ENTER @Datapath;Delta_x
2162        ENTER @Datapath;Datacount
2164        Hold_size=Datacount
2166      ELSE
2168        ENTER @Datapath;Datacount
2170        ENTER @Datapath;Hold_size
2172      END IF
2174      IF NOT Allocated THEN
2176        IF Datacount>=1 AND Hold_size>=1 THEN
2178          ALLOCATE Holding_file(Hold_size,2)
2180        ELSE
2182          ALLOCATE Holding_file(1,2)
2184        END IF
2186        Allocated=1
2188      END IF
2190      ENTER @Datapath;Holding_file(*)
2192      ASSIGN @Datapath TO *
2194      OFF ERROR
2196      IF NOT Flg THEN Delta_x=Holding_file(2,1)-Holding_file(1,1)
2198      IF Datacount=0 THEN Mistakeline
2200      !
2202      !Copy data from Holding_file(*) to Basket_file(*)
2204      !
2206      MAT Basket_file= (0.)
2208      IF Datacount>Basketsize THEN   !Receiving file too small.
2210        Allocated=0
2212        DEALLOCATE Holding_file(*)
2214        DISP " DATA FILE overflow, new data discarded. ";
2216        DISP " (continue) "
2218        BEEP
2220        PAUSE
```

```
2222          IF Status$ = "Y" THEN
2224             Curvecount = 0
2226             MAT Roster = (0)
2228          END IF
2230          Overflow = Hold_size
2232          GOTO Mistakelineset
2234       END IF
2236 Copydatafile:      !
2238       FOR R = 1 TO Datacount
2240          Basket_file(R,1) = Holding_file(R,1)
2242          Basket_file(R,2) = Holding_file(R,2)
2244       NEXT R
2246       Basketsize = Datacount
2248       GOTO Mistakeline
2250       !
2252 Mistakelineset:Datacount = 0
2254 Mistakeline:OFF KEY
2256       IF Allocated THEN DEALLOCATE Holding_file(*)
2258       LOOP
2260       EXIT IF TIMEDATE-Dtime > 1.8
2262       END LOOP
2264       DISP CHR$(12)
2266       OUTPUT 2 USING "#,K";"K"
2268       SUBEXIT
2270       !
2272       ! ////////////////////////////////////////////////////
2274       !
2276 Cant_findfile:      !Error in searching for the file.
2278       BEEP 500,.6
2280       SELECT ERRN
2282       CASE 56
2284          DISP "That file does not exist on this disk  ";
2286       CASE 72,73,76,82
2288          DISP Diskdrive$;" has failed or is not available ";
2290       CASE ELSE
2292          DISP ERRM$;
2294       END SELECT
2296       DISP " ....CONTINUE to try again."
2298       PAUSE
2300       Filename$ = ""
2302       Diskdrive$ = ""
2304       GOTO Selectdrive
2306       !
2308    SUBEND
2310    !
2312    ! ***************************************************************
2314    !
2316    SUB Data_check(Test$)
2318    !The following checks for a lower case "e" in an input number and,
2320    !if it exists, converts it to a number with an upper case "E". The
2322    !computer will recognize only upper case input otherwise.
2324       INTEGER Temp
2326       IF POS(Test$,"e") THEN
2328          Temp = POS(Test$,"e")
2330          Test$[Temp] = "E" &Test$[Temp + 1,LEN(Test$)]
2332       END IF
```

```
2334  !end of conversion.
2336      SUBEXIT
2338  SUBEND
2340  SUB Select_disk
2342 Select_disk: ! Original: 13 Nov 1984
2344              ! Revision: 02 Dec 1987
2346      OPTION BASE 1
2348      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
2350      COM /Interrupts/ INTEGER Intr_prty
2352      COM /Sys_msi/ Msi_id$
2354      COM /Sys/ Sys_id$
2356      INTEGER Local_prty,Dd,Pt,Choose(1)
2358      DIM Disc$(30)[60],Title$[40],Displ$[60]
2360      Local_prty=Intr_prty
2362      OFF KEY
2364      !
2366      ! Define the disk drives available for this system, reserve the
2368      ! first characters for the drive address and the characters after
2370      ! the - for a description of the drive.
2372      !
2374      ! Example:
2376      ! Disc$(1)=":,700,0,0      HP 9133H HARD disk, volume 0."
2378      !
2380      !
2382      Displ$=" SELECT DISK DRIVE  ... Abort will cancel. "
2384      Title$=" Available disk drives for this system. "
2386      Pt=1   ! allow only one select
2388      !
2390      IF Diskdrive$[1,1]<>":" THEN Diskdrive$=""
2392      IF Msi_id$[1,1]<>":" THEN Msi_id$=SYSTEM$("MSI")
2394      IF Msi_id$[1,1]<>":" THEN ! Must be HFS subdirectory
2396         Ms_path$=Msi_id$[1,POS(Msi_id$,":")-1]  ! strip off subdirs
2398         IF Ms_path$[LEN(Ms_path$);1]<>"/" THEN Ms_path$=Ms_path$&"/"
2400         Msi_id$=Msi_id$[POS(Msi_id$,":"),LEN(Msi_id$)]
2402      END IF
2404      Diskdrive$=TRIM$(Diskdrive$)
2406      Msi_id$=TRIM$(Msi_id$)
2408      IF LEN(Diskdrive$)>0 AND LEN(Msi_id$)>0 THEN
2410         Disc$(1)=Diskdrive$&RPT$(" ",17-LEN(Diskdrive$))
2412         Disc$(1)=Disc$(1)&"- Last selected disk drive."
2414         Dd=1
2416         IF Diskdrive$<>Msi_id$ THEN
2418            Disc$(2)=Msi_id$&RPT$(" ",17-LEN(Msi_id$))
2420            Disc$(2)=Disc$(2)&"- Start-up mass storage unit specifier."
2422            Dd=Dd+1
2424         ELSE
2426            Disc$(1)=Disc$(1)&"  Start-up MSUS."
2428         END IF
2430      ELSE
2432         IF LEN(Msi_id$)>0 THEN
2434            Disc$(1)=Msi_id$&RPT$(" ",17-LEN(Msi_id$))
2436            Disc$(1)=Disc$(1)&"- Start-up mass storage unit specifier."
2438            Dd=1
2440         ELSE
2442            Dd=0
2444         END IF
```

```
2446      END IF
2448 Disk: !
2450      ! ..... customize system drives here .....
2452      ! Follow format with - after unit specifier, description is
2454      ! optional but recommended.
2456      ! ..........................................................
2458      !
2460      Disc$(Dd+1)=":,702,0          - HP 9122 dual microfloppy left drive"
2462      Disc$(Dd+2)=":,702,1          - HP 9122 dual microfloppy right drive"
2464      Disc$(Dd+3)=":,703,0          - HP 9125 single 5.25 floppy drive"
2466      Disc$(Dd+4)=":,1400           - HP 9133H hard disk volume 1"
2468      !
2470      Dd=Dd+4    ! add the number of drive specifiers above
2472      !
2474      IF Sys_id$[1,4]<>"S300" THEN
2476         Disc$(Dd+1)=":,4,1           - LEFT internal series 200"
2478         Disc$(Dd+2)=":,4,0           - RIGHT internal series 200"
2480         Dd=Dd+2
2482      END IF
2484      ! ..........................................................
2486      !
2488      CALL Menu_scroll(Displ$,Title$,Disc$(*),Dd,Pt,Choose(*))
2490      IF Pt=0 THEN
2492         Diskdrive$="NO DISK"
2494      ELSE
2496         Dd=POS(Disc$(Choose(Pt)),"-")-1  ! find -
2498         IF Dd>5 THEN  !  valid msus
2500            Diskdrive$=TRIM$(Disc$(Choose(Pt))[1,Dd])
2502         ELSE
2504            DISP " ERROR in reading MSUS from string, - chr not found. "
2506            BEEP
2508            CALL Pause_key_on
2510            Diskdrive$="NO DISK"
2512         END IF
2514      END IF
2516 Diskselected:OFF KEY
2518      SUBEXIT
2520   SUBEND
2522   !
2524   ! ********************************************************
2526   !
2528   SUB Enterfilename(Ac$)
2530 Enterfilename:  ! Original: 13 Nov 1984
2532              ! Revision: 10 Dec 1990  includes HFS directories
2534      OPTION BASE 1
2536      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
2538      COM /Interrupts/ INTEGER Intr_prty
2540      INTEGER I,Ascii_num,Maskflag,Namelength
2542      DIM Test$[256],Hfs_temp$[161]
2544      Namelength=10
2546      IF LEN(Ms_path$)>0 THEN OUTPUT KBD USING "K,#";"#"&Ms_path$&"H"
2548      DISP " ENTER HFS directory PATH (no file)";
2550      IF Ac$<>"PATH" THEN
2552         DISP ", ENTER / for HFS ROOT or null for LIF...";
2554      END IF
2556      LINPUT Hfs_temp$
```

```
2558      Hfs_temp$=TRIM$(Hfs_temp$)
2560      IF LEN(Hfs_temp$)>0 THEN
2562         IF LEN(Hfs_temp$)>1 AND Hfs_temp$[LEN(Hfs_temp$);1]<>"/" THEN
2564            Hfs_temp$=Hfs_temp$&"/"
2566         END IF
2568         IF LEN(Hfs_temp$)=1 THEN Hfs_temp$=""
2570         Namelength=14
2572      END IF
2574      IF Ac$="PATH" THEN
2576         Ms_path$=Hfs_temp$
2578         SUBEXIT
2580      END IF
2582      IF LEN(Filename$)>0 THEN OUTPUT KBD USING "K,#";"#"&Filename$&"H"
2584 Efn: !
2586      DISP " ENTER the FILE NAME ... ";
2588      SELECT Ac$
2590      CASE "CAT"
2592         DISP "(ENTER CAT mask* or ENTER null to CAT)";
2594      CASE "ABORT"
2596         DISP "(ENTER null to ABORT) ";
2598      CASE "VALID"
2600         DISP "(must be a VALID name!) ";
2602      END SELECT
2604      LINPUT Test$
2606      Test$=TRIM$(Test$)
2608      IF LEN(Test$)=0 AND Ac$="VALID" THEN GOTO Enterfilename
2610      IF LEN(Test$)=0 THEN Abortline
2612      IF LEN(Test$)>Namelength THEN
2614         BEEP
2616         DISP "ERROR in NAME ENTRY - max ";Namelength;" chars, you have ";
2618         DISP LEN(Test$);" "
2620         WAIT 1.8
2622         OUTPUT 2 USING "K,#";"#"&Test$&"H"
2624         GOTO Efn
2626      END IF
2628      IF POS(Test$,"*")>1 THEN
2630         Test$=Test$[1,POS(Test$,"*")-1]
2632         Maskflag=1
2634      ELSE
2636         Maskflag=0
2638      END IF
2640      FOR I=1 TO LEN(Test$)
2642         Ascii_num=NUM(Test$[I])
2644         SELECT Ascii_num
2646         CASE 65 TO 90,95,97 TO 122,48 TO 57
2648            !Allowed characters
2650         CASE ELSE
2652            BEEP
2654            DISP "ERROR in NAME ENTRY--ILLEGAL CHARACTERS, TRY AGAIN."
2656            WAIT 1.8
2658            OUTPUT 2 USING "K,#";"#"&Test$&"H"
2660            GOTO Efn
2662         END SELECT
2664      NEXT I
2666      IF Maskflag THEN
2668         Filename$=Test$&"*"
```

```
2670      ELSE
2672         Filename$ = Test$
2674      END IF
2676      Ms_path$ = Hfs_temp$
2678      SUBEXIT
2680 Abortline:Filename$ = ""
2682      IF Ac$ = "CAT" THEN Ms_path$ = Hfs_temp$
2684      SUBEXIT
2686   SUBEND
2688   !
2690   ! ************************************************************
2692   !
```

```
100!   RE-STORE "GD_HISTOGRM:,1400"
102  !
104   SUB User_sub(REAL Wave(*),INTEGER D_volume,D_count)
106  !
108 Histogram_only:!
110       OPTION BASE 1
112       RAD
114       REAL Vmin,Vmax,V_last,Volt_z,Volt_100,V_first,Vptp
116       REAL Histogram(16384,2)
118       INTEGER No_o_bins,His_zero_lev
120       INTEGER His_100_lev,Maxpoint
122       REAL Delta_v,Delta_v_prc,Ov,Undr
124       !
126 Dateline:      !
128               !-----------------------------------
130               ! This program written by S.M. Chesnut
132               ! March 7, 1991
134               ! Last revision:
136               !-----------------------------------
138       INTEGER Indx,Min_bin,Half_bin,I,Level,Zipo,Hundred,Done,Auto,Pnts
140       DIM Ch$[1],X_units$[15],Y_units$[15]
142       ALLOCATE Hist_ary(16384)
144       Pnts=D_count
146       INPUT "What are the units of the x axis?",X_units$
148       INPUT "What are the units of the y axis?",Y_units$
150       GOSUB Mak_histogram
152       REDIM Histogram(No_o_bins,2),Wave(No_o_bins,2)
154       MAT Wave= Histogram
156       D_count=No_o_bins
158       DEALLOCATE Hist_ary(*)
160       SUBEXIT
162 Mak_histogram:   !
164               !
166       DISP "Calculating the histogram, please wait."
168       Auto=1
170       Done=0
172       Level=0
174       Ch$="n"
176       GOSUB Vmax_min_ptp
178       No_o_bins=1024
180       Min_bin=Pnts DIV 100
182       WHILE (Ch$="n") OR (Ch$="N")
184          WHILE NOT Done
186             Delta_v=Vptp/No_o_bins
188             Half_bin=No_o_bins DIV 2
190             MAT Hist_ary= (0)
192             FOR I=1 TO Pnts
194                Level=1+INT((Wave(I,2)-Vmin)/Delta_v)
196                IF Level>No_o_bins THEN
198                   Level=No_o_bins
200                END IF
202                Hist_ary(Level)=Hist_ary(Level)+1
204             NEXT I
206             Bug1=0
208             IF Bug1 THEN
```

```
210             PRINTER IS PRT
212             FOR I=1 TO No_o_bins
214                PRINT Hist_ary(I)
216             NEXT I
218             PRINTER IS CRT
220          END IF
222          Bug1=0
224          His_zero_lev=0
226          His_100_lev=0
228          Zipo=0
230          Hundred=0
232          FOR I=1 TO Half_bin
234             IF Hist_ary(I)>His_zero_lev THEN
236                His_zero_lev=Hist_ary(I)
238                Zipo=I
240             END IF
242          NEXT I
244          I=Half_bin
246          WHILE I<No_o_bins
248             IF Hist_ary(I+1)>His_100_lev THEN
250                His_100_lev=Hist_ary(I+1)
252                Hundred=I+1
254             END IF
256             I=I+1
258          END WHILE
260          IF Auto THEN
262             IF (His_zero_lev<Min_bin) THEN
264                No_o_bins=No_o_bins DIV 2
266                IF No_o_bins<128 THEN
268                   GOSUB Hist_message
270                   Done=1
272                END IF
274             ELSE
276                Done=1
278             END IF
280          ELSE
282             Done=1
284          END IF
286       END WHILE
288       GOSUB Calc_v_prms
290       GOSUB Hist_query
292    END WHILE
294    GOSUB Xy_histogram
296    RETURN
298 Hist_message:          !
300    PRINT "The number of bins in the histogram is less than 128."
302    PRINT "Therefore, the voltage resolution is quite bad and you"
304    PRINT "may find it is unacceptable. Keep this in mind when you"
306    PRINT "are asked if the histogram is an acceptable one."
308    WAIT 2.0
310    RETURN
312 !
314 Calc_v_prms: !
316    Volt_z=Vmin+Zipo*Delta_v-Delta_v/2.
318    Volt_100=Vmin+Hundred*Delta_v-Delta_v/2.
320    V_first=Wave(1,2)
322    V_last=Wave(Pnts,2)
324    Delta_v_prc=Delta_v*100/Vptp
```

```
326       RETURN
328  !
330 Hist_query: !
332  !
334       CLEAR SCREEN
336       PRINT "The first waveform point = ";V_first;" ";Y_units$;"."
338       PRINT
340       PRINT " The last waveform point = ";V_last;" ";Y_units$;"."
342       PRINT
344       PRINT "The minimum = ";Vmin;" ";Y_units$;"."
346       PRINT
348       PRINT "The maximum = ";Vmax;" ";Y_units$;"."
350       PRINT
352       PRINT "There were ";No_o_bins;"bins used in the histogram."
354       PRINT
356       PRINT "Each histogram bin is equivalent to";Delta_v;" ";Y_units$;","
358       PRINT
360       PRINT "or";Delta_v_prc;"% of the waveform peak-to-peak."
362       PRINT
364       PRINT "The 0% level occurs at";Volt_z;" ";Y_units$;" with";His_zero_lev;"occurrences."
366       PRINT
368       PRINT "The 100% level occurs at";Volt_100;" ";Y_units$;" with";His_100_lev;"occurrences."
370       INPUT "Is this an acceptable histogram?",Ch$
372       IF (Ch$="n") OR (Ch$="N") THEN
374          INPUT "How many histogram bins would you like to use?",No_o_bins
376          Done=0
378          Auto=0
380       END IF
382       CLEAR SCREEN
384       RETURN
386 Xy_histogram:       !
388       Range=Wave(Pnts,1)-Wave(1,1)
390       IF Range>1.E-50 THEN
392          Base=10^INT(LGT(Range))
394          SELECT Range
396          CASE <=2*Base
398             Factor=Base/5
400          CASE <=5*Base
402             Factor=Base/2
404          CASE <=10*Base
406             Factor=Base
408          END SELECT
410       ELSE
412          Factor=1
414       END IF
416       Xmin=Factor*(INT(Wave(1,1)/Factor))
418       Xmax=Factor*(INT(Wave(Pnts,1)/Factor))
420       Edge_lt=Xmin-.55*(Xmax-Xmin)
422       Edge_rt=Xmin-.35*(Xmax-Xmin)
424       Delta_x=(Edge_rt-Edge_lt)/MAX(Hist_ary(*))
426       Coeff=REAL(No_o_bins/(No_o_bins-1))
428       Coeff2=REAL(MAX(Hist_ary(*))/(MAX(Hist_ary(*))-1))
430       FOR I=1 TO No_o_bins
432          Histogram(I,1)=Hist_ary(I)*Coeff2*Delta_x+Edge_lt
434          Histogram(I,2)=(I-1)*Coeff*Delta_v+Vmin
436       NEXT I
```

```
438       RETURN
440       !
442       ! ////////////////////////////////////////////////////////
444       !
446 Vmax_min_ptp: !
448       Vmax=Wave(1,2)
450       Vmin=Wave(1,2)
452       FOR I=1 TO Pnts
454          IF Wave(I,2)<Vmin THEN Vmin=Wave(I,2)
456          IF Wave(I,2)>Vmax THEN
458             Vmax=Wave(I,2)
460             Maxpoint=I
462          END IF
464       NEXT I
466       Vptp=Vmax-Vmin
468       RETURN
470    SUBEND
472 !
474 !*************************************************************************
476 !
```

```
100  ! RE-STORE "MATH_OPS:,1400"
102   COM /Sys_msi/ Msi_id$[20]
104   COM /Sys/ Sys_id$[10]
106   COM /Interrupts/ INTEGER Intr_prty
108   !
110   OUTPUT KBD USING "K,#";"SCRATCH KEYE"        !ERASE SOFT KEYS
112   CONTROL KBD,15;0! sets the color of the soft keys
114   CONTROL KBD,2;1
116   Intr_prty=1
118   CALL Do_op
120   OUTPUT KBD USING "K,#";"LOAD KEYE"! restore the typing aid keys
122   PRINT TABXY(1,5);"END of program. So long."
124   MASS STORAGE IS ":,1400"
126   !
128   END
130   !
132   !-------------------------------------------------------
134   ! This program performs simple math operations on waveform data.
136   ! You may add, subtract, multiply, or divide (non-zero) the data
138   ! by a constant or you may integrate, differentiate, or time shift
140   ! the data.
142   !-------------------------------------------------------
144   !
146 Date_line:   !
148           ! S. M. Chesnut
150           ! May 21,1991
152           !
154   SUB Do_op
156   !
158      OPTION BASE 1
160      DEG
162      COM /Flgs/ Stp_flg
164      COM /Interrupts/ INTEGER Intr_prty
166      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
168      COM /Data_stuff/ INTEGER Number,REAL Delta_x,REAL Strt_time
170      COM /Data_vars/ REAL Wave(4096,2),New_wave(4096,2),INTEGER Loaded
172   !
174      INTEGER Local_prty,Basketsize
176      DIM Data_id$[40],Test$[20],Ch$[1]
178      REAL Last_pnt,Wave_int,Time_plc
180 Do_op:     !
182      OFF KEY
184      OFF KNOB
186      OFF KBD
188      Interrupted=1
190      Local_prty=Intr_prty
192         !
194      Basketsize=4096
196      Filename$=""
198      Diskdrive$=""
200      Loaded=0
202      Number=0
204      LOOP
206         IF Interrupted THEN GOSUB Menu
208         ON KEY 9 LABEL "EXIT        ",Local_prty+3 GOTO Ret
```

```
210       END LOOP
212 Ret:      !
214       OFF KEY
216       CLEAR SCREEN
218       Stp_flg=1
220       SUBEXIT
222       !
224 Menu:     !
226       CLEAR SCREEN
228       OFF KEY
230       OFF KBD
232       OFF KNOB
234       Interrupted=0
236       DISP "Select the appropriate soft key for the desired operation."
238       ON KEY 0 LABEL "INTEGRATE",Local_prty+1 GOSUB Integrate
240       ON KEY 2 LABEL "DIFFERENTIATE",Local_prty+1 GOSUB Differentiate
242       ON KEY 4 LABEL "LOAD  FILE",Local_prty+1 GOSUB Load_data
244       ON KEY 6 LABEL "TIME  SHIFT",Local_prty+1 GOSUB Cal_start_point
246       ON KEY 8 LABEL "Y +*/- CONST",Local_prty+2 GOSUB Call_const_math
248       RETURN
250       !
252       !*****************************************************************
254       !
256 Integrate: !
258       IF NOT Loaded THEN
260          BEEP
262          DISP "NO FILE IN MEMORY, PLEASE LOAD A FILE FIRST."
264          WAIT 1.5
266          RETURN
268          CLEAR SCREEN
270       END IF
272       OFF KEY
274       OFF KBD
276       OFF KNOB
278       CLEAR SCREEN
280       Interrupted=1
282       Time_plc=0
284       Last_pnt=Wave(1,2)
286       New_wave(1,2)=0
288       FOR I=2 TO Number
290          Wave_int=.5*Delta_x*(Last_pnt+Wave(I,2))+New_wave(I-1,2)
292          Last_pnt=Wave(I,2)
294          New_wave(I,2)=Wave_int
296          New_wave(I,1)=Time_plc
298          Time_plc=Time_plc+Delta_x
300       NEXT I
302       CALL Store_new
304       RETURN
306       !
308 !
310 !*************************************************************************
312 !
314 Differentiate: !
316       IF NOT Loaded THEN
318          BEEP
320          DISP "NO FILE IN MEMORY, PLEASE LOAD A FILE FIRST."
322          WAIT 1.5
324          RETURN
```

```
326         CLEAR SCREEN
328       END IF
330       OFF KEY
332       OFF KBD
334       OFF KNOB
336       Interrupted=1
338       Time_plc=Delta_x
340       New_wave(1,2)=0
342       New_wave(1,1)=0
344       FOR I=2 TO Number
346          New_wave(I,2)=(Wave(I,2)-Wave(I-1,2))/Delta_x
348          New_wave(I,1)=Time_plc
350          Time_plc=Time_plc+Delta_x
352       NEXT I
354       CALL Store_new
356       RETURN
358       !
360       !*****************************************************************
362       !
364 Load_data:       !
366       Loaded=1
368       Intr_prty=Local_prty+2
370       CALL Load_disk_data(Wave(*),Basketsize,Data_id$,0)
372       Intr_prty=Local_prty
374       IF Number=0 THEN
376          BEEP
378          DISP "NO FILE WAS READ, PLEASE TRY AGAIN."
380          WAIT 1.5
382          GOTO Load_data
384       END IF
386       REDIM Wave(Number,2)
388       REDIM New_wave(Number,2)
390       RETURN
392       !
394       !*****************************************************************
396       !
398 Cal_start_point:       !
400       Intr_prty=Local_prty+2
402       CALL Start_point
404       Intr_prty=Local_prty
406       RETURN
408       !
410       !*****************************************************************
412       !
414 Call_const_math:       !
416       Intr_prty=Local_prty+2
418       CALL Const_math
420       Intr_prty=Local_prty
422       RETURN
424    SUBEND
426    !
428    !*****************************************************************
430    !
432    SUB Start_point
434    !
436       OPTION BASE 1
```

```
438       DEG
440       COM /Flgs/ Stp_flg
442       COM /Interrupts/ INTEGER Intr_prty
444       COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
446       COM /Data_vars/ REAL Wave(4096,2),New_wave(4096,2),INTEGER Loaded
448       COM /Data_stuff/ INTEGER Number,REAL Delta_x,REAL Strt_time
450   !
452       INTEGER Local_prty,Indx,Pointer
454       REAL Max_pnt,Min_pnt
456 Start_point:     !
458       PRINT SYSTEM$("SYSTEM PRIORITY")
460       OFF KEY
462       OFF KNOB
464       OFF KBD
466       Interrupted=1
468       Local_prty=Intr_prty
470       IF NOT Loaded THEN
472          BEEP
474          DISP "THERE IS NO FILE IN MEMORY, PLEASE LOAD FIRST."
476          SUBEXIT
478       END IF
480       LOOP
482          IF Interrupted THEN GOSUB Shiftmenu
484          ON KEY 9 LABEL "EXIT       ",Local_prty+5 GOTO Ret
486       END LOOP
488 Ret:       !
490       OFF KEY
492       CLEAR SCREEN
494       Stp_flg=1
496       SUBEXIT
498 Shiftmenu:       !
500       Interrupted=0
502       CLEAR SCREEN
504       OFF KEY
506       OFF KBD
508       OFF KNOB
510       DISP "Select the appropriate soft key for the new t=0 point."
512       ON KEY 0 LABEL "Max value",Local_prty+1 GOTO Maxval
514       ON KEY 2 LABEL "Min value",Local_prty+1 GOTO Minval
516       ON KEY 4 LABEL "Keyboard input",Local_prty+1 GOTO Key_in
518       RETURN
520 Maxval:         !
522       OFF KEY
524       OFF KBD
526       OFF KNOB
528       CLEAR SCREEN
530       Max_pnt=-1.0E+60
532       FOR I=1 TO Number
534          IF Wave(I,2)>Max_pnt THEN
536             Max_pnt=Wave(I,2)
538             Indx=I
540          END IF
542       NEXT I
544       GOSUB Reorder
546       CALL Store_new
548       SUBEXIT
```

```
550 Minval:       !
552       OFF KEY
554       OFF KBD
556       OFF KNOB
558       Min_pnt=1.E+60
560       FOR I=1 TO Number
562          IF Wave(I,2)<Min_pnt THEN
564             Min_pnt=Wave(I,2)
566             Indx=I
568          END IF
570       NEXT I
572       GOSUB Reorder
574       CALL Store_new
576       SUBEXIT
578 Key_in:            !
580       OFF KEY
582       OFF KBD
584       OFF KNOB
586       INPUT "Enter the new starting point index",Indx
588       GOSUB Reorder
590       CALL Store_new
592       SUBEXIT
594 Reorder:       !
596       FOR I=1 TO Number
598          Pointer=Indx+I-1
600          IF Pointer>Number THEN Pointer=Pointer-Number
602          New_wave(I,2)=Wave(Pointer,2)
604          New_wave(I,1)=Wave(I,1)
606       NEXT I
608       RETURN
610    SUBEND
612    !
614    !*****************************************************************
616    !
618    SUB Store_new
620    !
622       OPTION BASE 1
624       DEG
626       COM /Interrupts/ INTEGER Intr_prty
628       COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
630       COM /Data_vars/ REAL Wave(4096,2),New_wave(4096,2),INTEGER Loaded
632       COM /Data_stuff/ INTEGER Number,REAL Delta_x,REAL Strt_time
634    !
636       INTEGER Local_prty
638    !
640 Store_new:       !
642       Local_prty=Intr_prty
644       Filename$=""
646       Diskdrive$=""
648       INPUT "Enter a 40 char. (or less) data description.",Data_id$
650       Intr_prty=Local_prty+1
652       CALL Data_to_disk_r(1,Number,New_wave(*),Data_id$)
654       Intr_prty=Local_prty
656       Loaded=0
658       SUBEXIT
660    SUBEND
```

```
662   !
664   !*****************************************************************
666   !
668   SUB Const_math
670   !
672       OPTION BASE 1
674       DEG
676       COM /Flgs/ Stp_flg
678       COM /Interrupts/ INTEGER Intr_prty
680       COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
682       COM /Data_stuff/ INTEGER Number,REAL Delta_x,REAL Strt_time
684       COM /Data_vars/ REAL Wave(4096,2),New_wave(4096,2),INTEGER Loaded
686   !
688       INTEGER Local_prty
690       REAL Last_pnt,Wave_int,Time_plc
692 Const_math:    !
694       OFF KEY
696       OFF KNOB
698       OFF KBD
700       Interrupted=1
702       Local_prty=Intr_prty
704       IF NOT Loaded THEN
706          BEEP
708          DISP "THERE IS NO FILE IN MEMORY, PLEASE LOAD FIRST."
710          SUBEXIT
712       END IF
714       LOOP
716          IF Interrupted THEN GOSUB Const_menu
718          ON KEY 9 LABEL "EXIT        ",Local_prty+3 GOTO Ret
720       END LOOP
722 Ret:       !
724       OFF KEY
726       CLEAR SCREEN
728       Stp_flg=1
730       SUBEXIT
732       !
734 Const_menu:       !
736       Interrupted=0
738       CLEAR SCREEN
740       OFF KEY
742       OFF KBD
744       OFF KNOB
746       DISP "Select the appropriate soft key for the desired operation."
748       ON KEY 0 LABEL "ADD",Local_prty+1 GOTO Add
750       ON KEY 2 LABEL "SUBTRACT",Local_prty+1 GOTO Sub
752       ON KEY 4 LABEL "MULTIPLY",Local_prty+1 GOTO Mult
754       ON KEY 6 LABEL "DIVIDE",Local_prty+1 GOTO Divide
756       RETURN
758       !
760       !
762 Add:       !
764       OFF KEY
766       OFF KBD
768       OFF KNOB
770       CLEAR SCREEN
772       GOSUB Get_const
```

```
774       FOR I=1 TO Number
776          Wave(I,2)=Wave(I,2)+Const
778       NEXT I
780       CALL Store_new
782       SUBEXIT
784 Sub:          !
786       OFF KEY
788       OFF KBD
790       OFF KNOB
792       CLEAR SCREEN
794       GOSUB Get_const
796       FOR I=1 TO Number
798          Wave(I,2)=Wave(I,2)-Const
800       NEXT I
802       CALL Store_new
804       SUBEXIT
806 Mult:         !
808       OFF KEY
810       OFF KBD
812       OFF KNOB
814       CLEAR SCREEN
816       GOSUB Get_const
818       FOR I=1 TO Number
820          Wave(I,2)=Wave(I,2)*Const
822       NEXT I
824       CALL Store_new
826       SUBEXIT
828 Div_err:       !
830       BEEP
832       DISP "DIVISION BY ZERO IS NOT ALLOWED."
834       WAIT 1.0
836       DISP "Input a new constant."
838       WAIT 1.0
840       GOSUB Get_const
842 Divide:         !
844       OFF KEY
846       OFF KBD
848       OFF KNOB
850       CLEAR SCREEN
852       GOSUB Get_const
854       IF Const<>0 THEN
856          FOR I=1 TO Number
858             Wave(I,2)=Wave(I,2)/Const
860          NEXT I
862          CALL Store_new
864          SUBEXIT
866       ELSE
868          GOTO Div_err
870       END IF
872 Const_error:      !
874       CLEAR SCREEN
876       BEEP
878       DISP "ERROR IN CONSTANT INPUT, try again."
880       WAIT 1.0
882 Get_const:      !
884       CLEAR SCREEN
```

```
886       ON ERROR GOTO Const_error
888       INPUT "What is the value of the constant?",Test$
890       Const=VAL(Test$)
892       OFF ERROR
894       RETURN
896       !
898     SUBEND
900     !
902     !*************************************************************************
904     !
906     SUB Load_disk_data(Basket_file(*),INTEGER Basketsize,Data_id$,INTEGER Flg)
908   Load_disk_data:   ! Original: 13 Nov 1984
910                     ! Revision: 02 Dec 1987
912       !This routine will enter data files from the disk
914       OPTION BASE 1
916       !
918       COM /Sys/ Sys_id$
920       COM /History/ Status$[1],Time_orgn$[8],Date_orgn$[11]
922       COM /History/ Time_chng$[8],Date_chng$[11],Description$[160]
924     !
926       COM /Labels/ Labels$(30)[60],INTEGER Lbl_count,REAL Lbl_addr(30,6)
928     !Lbl_addr: x, y, pen, size, LDIR, LORG
930     !
932       COM /Data_param/ INTEGER Datacount,Filesize,Curvecount,Roster(17,4)
934       COM /Data_param/ REAL Sym_size,Symbol$(17)[2],Curve_id$(17)[40]
936       COM /Data_param/ REAL Xmin_data,Xmax_data
938       COM /Data_param/ REAL Ymin_data,Ymax_data
940     !
942     !Roster: Curve#, Start Addr in File(*), Datacount, and PEN
944     !Symbol$(i)="" or "Y" => no symbol, connect pts
946     !Symbol$(i)="*Y" => * symbol, connect pts
948     !Symbol$(i)="*N" => * symbol, do not connect pts
950     !
952       COM /Background/ Graphtype$[12],Margins$(2)[10],Papersize$[1]
954       COM /Background/ REAL Pen_speed,INTEGER Backgnd_pen,Auto_time
956       COM /Background/ INTEGER Auto_file,REAL X_cross_y,Y_cross_x
958       COM /Background/ Xgrid_tick$[4],INTEGER Xmajor,Xminor
960       COM /Background/ Ygrid_tick$[4],INTEGER Ymajor,Yminor
962       COM /Background/ REAL Xmin_graph,Xmax_graph,Ymin_graph,Ymax_graph
964     !
966       COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
968       COM /Interrupts/ INTEGER Intr_prty
970       COM /Enlarge_file/ INTEGER Overflow
972       COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
974       COM /Data_stuff/ INTEGER Number,REAL Delta_x,REAL Strt_time
976     !
978       INTEGER R,Hold_size,Local_prty,Allocated,Fls_cnt
980       DIM Ac$[5],Tempfile$[10],Mask$[10],Ftype$[5],Fls$(1)[14]
982       REAL Dtime
984       OFF KEY
986       Local_prty=Intr_prty
988       !
990       !Select the disk drive where the data exists
992       !
994       IF Overflow<>0 THEN Overflow=0
996       Hold_size=0
```

```
998       Dtime=0.
1000      Allocated=0
1002 Selectdrive:      !
1004      IF Diskdrive$="NO DISK" THEN Diskdrive$=""
1006      IF LEN(Diskdrive$)>0 THEN GOTO Choosefilename
1008      GRAPHICS OFF
1010      OUTPUT 2 USING "#,K";"K"
1012      CALL Select_disk
1014      IF Diskdrive$="NO DISK" THEN GOTO Mistakelineset
1016 Choosefilename: !
1018      Tempfile$=Filename$
1020      IF LEN(Filename$)>0 THEN GOTO Bring_in_data
1022      Ac$="CAT"
1024      CALL Enterfilename(Ac$)
1026      IF LEN(Filename$)=0 OR POS(Filename$,"*")>1 THEN
1028         IF POS(Filename$,"*")>1 THEN ! set mask$
1030            Mask$=Filename$[1,POS(Filename$,"*")-1]
1032            Filename$=""
1034         ELSE
1036            Mask$=""! no preselection
1038         END IF
1040         Ftype$="BDAT "  ! examine BDAT files only
1042         Fls_cnt=1  ! select one file
1044         Intr_prty=Local_prty+1
1046         CALL File_menu(Mask$,Ftype$,Fls$(*),Fls_cnt,0,0)
1048         Intr_prty=Local_prty
1050         Filename$=Fls$(1)
1052         IF LEN(Filename$)=0 THEN  ! aborted
1054            Filename$=Tempfile$
1056            GOTO Mistakelineset
1058         END IF
1060      END IF
1062 Bring_in_data: !
1064      !
1066      !Find this file on the disk.
1068      !
1070      ON ERROR GOTO Cant_findfile
1072      ASSIGN @Datapath TO Filename$&Diskdrive$
1074      OFF ERROR
1076      Dtime=TIMEDATE
1078      DISP " LOADING disk file: ";Filename$;" ... ";
1080      ON ERROR GOTO Bad_file
1082      ENTER @Datapath;Status$
1084      OFF ERROR
1086      ON ERROR GOTO Cant_findfile
1088      SELECT Status$
1090      CASE "Y"    ! All graphics/data parameters exist.REN 100,2
1092         DISP " Complete graph. "
1094         ENTER @Datapath;Time_orgn$,Date_orgn$
1096         ENTER @Datapath;Time_chng$,Date_chng$
1098         ENTER @Datapath;Description$
1100         ENTER @Datapath;Labels$(*),Lbl_count,Lbl_addr(*)
1102         ENTER @Datapath;Curve_id$(*),Symbol$(*)
1104         ENTER @Datapath;Roster(*),Curvecount
1106         ENTER @Datapath;Graphtype$,Margins$(*)
1108         ENTER @Datapath;X_cross_y,Y_cross_x
```

```
1110        ENTER @Datapath;Xgrid_tick$,Xmajor,Xminor
1112        ENTER @Datapath;Ygrid_tick$,Ymajor,Yminor
1114        ENTER @Datapath;Xmin_graph,Xmax_graph
1116        ENTER @Datapath;Ymin_graph,Ymax_graph
1118      CASE "N"     ! Only data parameters exist.
1120        DISP " RAW data. "
1122      CASE ELSE
1124 Bad_file: DISP CHR$(12)
1126        DISP "Data file is not recognized, entry aborted.";
1128        DISP " ...continue."
1130        BEEP
1132        PAUSE
1134        OFF ERROR
1136        GOTO Mistakelineset
1138      END SELECT
1140      !
1142      ENTER @Datapath;Data_id$
1144      IF Flg THEN
1146        ENTER @Datapath;Delta_x
1148        ENTER @Datapath;Datacount
1150        Hold_size=Datacount
1152      ELSE
1154        ENTER @Datapath;Datacount
1156        ENTER @Datapath;Hold_size
1158      END IF
1160      IF NOT Allocated THEN
1162        IF Datacount>=1 AND Hold_size>=1 THEN
1164          ALLOCATE Holding_file(Hold_size,2)
1166        ELSE
1168          ALLOCATE Holding_file(1,2)
1170        END IF
1172        Allocated=1
1174      END IF
1176      ENTER @Datapath;Holding_file(*)
1178      ASSIGN @Datapath TO *
1180      OFF ERROR
1182      IF NOT Flg THEN
1184        Delta_x=Holding_file(2,1)-Holding_file(1,1)
1186        Strt_time=Holding_file(1,1)
1188      END IF
1190      IF Datacount=0 THEN Mistakeline
1192      !
1194      !Copy data from Holding_file(*) to Basket_file(*)
1196      !
1198      MAT Basket_file= (0.)
1200      IF Datacount>Basketsize THEN   !Receiving file too small.
1202        Allocated=0
1204        DEALLOCATE Holding_file(*)
1206        DISP " DATA FILE overflow, new data discarded. ";
1208        DISP " (continue) "
1210        BEEP
1212        PAUSE
1214        IF Status$="Y" THEN
1216          Curvecount=0
1218          MAT Roster= (0)
1220        END IF
```

```
1222          Overflow=Hold_size
1224          GOTO Mistakelineset
1226       END IF
1228 Copydatafile:      !
1230       FOR R=1 TO Datacount
1232          Basket_file(R,1)=Holding_file(R,1)
1234          Basket_file(R,2)=Holding_file(R,2)
1236       NEXT R
1238       Number=Datacount
1240       GOTO Mistakeline
1242       !
1244 Mistakelineset:Datacount=0
1246 Mistakeline:OFF KEY
1248       IF Allocated THEN DEALLOCATE Holding_file(*)
1250       LOOP
1252       EXIT IF TIMEDATE-Dtime>1.8
1254       END LOOP
1256       DISP CHR$(12)
1258       OUTPUT 2 USING "#,K";"K"
1260       SUBEXIT
1262       !
1264       ! ////////////////////////////////////////////////////
1266       !
1268 Cant_findfile:      !Error in searching for the file.
1270       BEEP 500,.6
1272       SELECT ERRN
1274       CASE 56
1276          DISP "That file does not exist on this disk  ";
1278       CASE 72,73,76,82
1280          DISP Diskdrive$;" has failed or is not available ";
1282       CASE ELSE
1284          DISP ERRM$;
1286       END SELECT
1288       DISP " ....CONTINUE to try again."
1290       PAUSE
1292       Filename$=""
1294       Diskdrive$=""
1296       GOTO Selectdrive
1298       !
1300    SUBEND
1302    !
1304    ! ***************************************************************
1306    !
1308    SUB Data_to_disk_r(INTEGER Curve,Datacount,REAL Basket_file(*),Data_id$)
1310 Data_to_disk_r:  ! Original: 13 Nov 1984
1312                  ! Revision: 02 Dec 1987
1314       !This routine will SAVE data files on the disk in RAW data format.
1316       OPTION BASE 1
1318       COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
1320       COM /Interrupts/ INTEGER Intr_prty
1322       INTEGER Local_prty,Diskspace
1324       DIM Ac$[5],Status$[1],Tempfile$[14]
1326       REAL Dtime
1328       OFF KEY
1330       Local_prty=Intr_prty
1332       Dtime=0.
```

```
1334      !
1336      !Select the disk drive for data storage
1338      !
1340 Selectdrive: !
1342      GRAPHICS OFF
1344      OUTPUT 2 USING "#,K";"K"
1346      CALL Select_disk
1348      IF Diskdrive$="NO DISK" THEN GOTO Mistakeline
1350 Choosefilename: !
1352      Ac$="ABORT"
1354      Tempfile$=Filename$
1356      CALL Enterfilename(Ac$)
1358      IF LEN(Filename$)=0 THEN
1360         Filename$=Tempfile$
1362         GOTO Mistakeline
1364      END IF
1366 Send_to_disk: ! Create file and save information.
1368      ON ERROR GOTO Cant_savedata
1370      Diskspace=INT((Datacount*16.0)/256)+2
1372      CREATE BDAT Ms_path$&Filename$&Diskdrive$,Diskspace,256
1374      Dtime=TIMEDATE
1376      DISP " SAVING data for CURVE # ";Curve;". "
1378      Status$="N"
1380      ASSIGN @Datapath TO Ms_path$&Filename$&Diskdrive$
1382      OUTPUT @Datapath;Status$
1384      OUTPUT @Datapath;Data_id$    !40 chrs description if single curve.
1386      OUTPUT @Datapath;Datacount   !number of xy points
1388      OUTPUT @Datapath;Datacount   !size of array (same as above)
1390      OUTPUT @Datapath;Basket_file(*)
1392      ASSIGN @Datapath TO *
1394      OFF ERROR
1396      !
1398 Mistakeline:OFF KEY
1400      LOOP
1402      EXIT IF TIMEDATE-Dtime>1.8
1404      END LOOP
1406      DISP CHR$(12)
1408      OUTPUT 2 USING "#,K";"K"
1410      SUBEXIT
1412      !
1414      ! ////////////////////////////////////////////////////
1416      !
1418 Cant_savedata:  !
1420      BEEP 500,.6
1422      SELECT ERRN
1424      CASE 72,73,76,78,81,82,90,93
1426         DISP Diskdrive$;" has failed or is not available ";
1428         DISP " ....CONTINUE to try again."
1430         CALL Pause_key_on
1432         Filename$=Tempfile$
1434      CASE 84,85
1436         DISP " This disk is not initialized ";
1438         DISP " ....CONTINUE to try again."
1440         CALL Pause_key_on
1442         Filename$=Tempfile$
1444      CASE 55,64
```

```
1446        DISP " This disk is full, insert new floppy and/or";
1448        DISP " select new drive ...CONTINUE "
1450        CALL Pause_key_on
1452        Filename$=Tempfile$
1454      CASE ELSE
1456        CALL Errortrap
1458        GOTO Send_to_disk
1460      END SELECT
1462      GOTO Selectdrive
1464      !
1466    SUBEND
1468    !
1470    ! ********************************************************************
1472    !
1474    SUB Select_disk
1476  Select_disk: ! Original: 13 Nov 1984
1478            ! Revision: 02 Dec 1987
1480      OPTION BASE 1
1482      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
1484      COM /Interrupts/ INTEGER Intr_prty
1486      COM /Sys_msi/ Msi_id$
1488      COM /Sys/ Sys_id$
1490      INTEGER Local_prty,Dd,Pt,Choose(1)
1492      DIM Disc$(30)[60],Title$[40],Displ$[60]
1494      Local_prty=Intr_prty
1496      OFF KEY
1498      !
1500      ! Define the disk drives available for this system, reserve the
1502      ! first characters for the drive address and the characters after
1504      ! the - for a description of the drive.
1506      !
1508      ! Example:
1510      ! Disc$(1)=":,700,0,0      HP 9133H HARD disk, volume 0."
1512      !
1514      !
1516      Displ$=" SELECT DISK DRIVE  ... Abort will cancel. "
1518      Title$=" Available disk drives for this system. "
1520      Pt=1   ! allow only one select
1522      !
1524      IF Diskdrive$[1,1]<>":" THEN Diskdrive$=""
1526      IF Msi_id$[1,1]<>":" THEN Msi_id$=SYSTEM$("MSI")
1528      IF Msi_id$[1,1]<>":" THEN ! Must be HFS subdirectory
1530        Ms_path$=Msi_id$[1,POS(Msi_id$,":")-1]  ! strip off subdirs
1532        IF Ms_path$[LEN(Ms_path$);1]<>"/" THEN Ms_path$=Ms_path$&"/"
1534        Msi_id$=Msi_id$[POS(Msi_id$,":"),LEN(Msi_id$)]
1536      END IF
1538      Diskdrive$=TRIM$(Diskdrive$)
1540      Msi_id$=TRIM$(Msi_id$)
1542      IF LEN(Diskdrive$)>0 AND LEN(Msi_id$)>0 THEN
1544        Disc$(1)=Diskdrive$&RPT$(" ",17-LEN(Diskdrive$))
1546        Disc$(1)=Disc$(1)&"- Last selected disk drive."
1548        Dd=1
1550        IF Diskdrive$<>Msi_id$ THEN
1552          Disc$(2)=Msi_id$&RPT$(" ",17-LEN(Msi_id$))
1554          Disc$(2)=Disc$(2)&"- Start-up mass storage unit specifier."
1556          Dd=Dd+1
```

```
1558        ELSE
1560           Disc$(1)=Disc$(1)&"  Start-up MSUS."
1562        END IF
1564     ELSE
1566        IF LEN(Msi_id$)>0 THEN
1568           Disc$(1)=Msi_id$&RPT$(" ",17-LEN(Msi_id$))
1570           Disc$(1)=Disc$(1)&"- Start-up mass storage unit specifier."
1572           Dd=1
1574        ELSE
1576           Dd=0
1578        END IF
1580     END IF
1582 Disk: !
1584     ! ..... customize system drives here .....
1586     ! Follow format with - after unit specifier, description is
1588     ! optional but recommended.
1590     ! ..........................................................
1592     !
1594     Disc$(Dd+1)=":,702,0          - HP 9122 dual microfloppy left drive"
1596     Disc$(Dd+2)=":,702,1          - HP 9122 dual microfloppy right drive"
1598     Disc$(Dd+3)=":,703,0          - HP 9125 single 5.25 floppy drive"
1600     Disc$(Dd+4)=":,1400           - HP 9133H hard disk volume 1"
1602     !
1604     Dd=Dd+4    ! add the number of drive specifiers above
1606     !
1608     IF Sys_id$[1,4]<>"S300" THEN
1610        Disc$(Dd+1)=":,4,1            - LEFT internal series 200"
1612        Disc$(Dd+2)=":,4,0            - RIGHT internal series 200"
1614        Dd=Dd+2
1616     END IF
1618     ! ..........................................................
1620     !
1622     CALL Menu_scroll(Displ$,Title$,Disc$(*),Dd,Pt,Choose(*))
1624     IF Pt=0 THEN
1626        Diskdrive$="NO DISK"
1628     ELSE
1630        Dd=POS(Disc$(Choose(Pt)),"-")-1  ! find -
1632        IF Dd>5 THEN  ! valid msus
1634           Diskdrive$=TRIM$(Disc$(Choose(Pt))[1,Dd])
1636        ELSE
1638           DISP " ERROR in reading MSUS from string, - chr not found. "
1640           BEEP
1642           CALL Pause_key_on
1644           Diskdrive$="NO DISK"
1646        END IF
1648     END IF
1650 Diskselected:OFF KEY
1652     SUBEXIT
1654  SUBEND
1656  !
1658  ! ********************************************************
1660  !
1662  SUB Enterfilename(Ac$)
1664 Enterfilename:  ! Original: 13 Nov 1984
1666              ! Revision: 10 Dec 1990  includes HFS directories
1668     OPTION BASE 1
```

```
1670      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
1672      COM /Interrupts/ INTEGER Intr_prty
1674      INTEGER I,Ascii_num,Maskflag,Namelength
1676      DIM Test$[256],Hfs_temp$[161]
1678      Namelength=10
1680      IF LEN(Ms_path$)>0 THEN OUTPUT KBD USING "K,#";"#"&Ms_path$&"H"
1682      DISP " ENTER HFS directory PATH (no file)";
1684      IF Ac$<>"PATH" THEN
1686         DISP ", ENTER / for HFS ROOT or null for LIF...";
1688      END IF
1690      LINPUT Hfs_temp$
1692      Hfs_temp$=TRIM$(Hfs_temp$)
1694      IF LEN(Hfs_temp$)>0 THEN
1696         IF LEN(Hfs_temp$)>1 AND Hfs_temp$[LEN(Hfs_temp$);1]<>"/" THEN
1698            Hfs_temp$=Hfs_temp$&"/"
1700         END IF
1702         IF LEN(Hfs_temp$)=1 THEN Hfs_temp$=""
1704         Namelength=14
1706      END IF
1708      IF Ac$="PATH" THEN
1710         Ms_path$=Hfs_temp$
1712         SUBEXIT
1714      END IF
1716      IF LEN(Filename$)>0 THEN OUTPUT KBD USING "K,#";"#"&Filename$&"H"
1718 Efn: !
1720      DISP " ENTER the FILE NAME ... ";
1722      SELECT Ac$
1724      CASE "CAT"
1726         DISP "(ENTER CAT mask* or ENTER null to CAT)";
1728      CASE "ABORT"
1730         DISP "(ENTER null to ABORT) ";
1732      CASE "VALID"
1734         DISP "(must be a VALID name!) ";
1736      END SELECT
1738      LINPUT Test$
1740      Test$=TRIM$(Test$)
1742      IF LEN(Test$)=0 AND Ac$="VALID" THEN GOTO Enterfilename
1744      IF LEN(Test$)=0 THEN Abortline
1746      IF LEN(Test$)>Namelength THEN
1748         BEEP
1750         DISP "ERROR in NAME ENTRY - max ";Namelength;" chars, you have ";
1752         DISP LEN(Test$);" "
1754         WAIT 1.8
1756         OUTPUT 2 USING "K,#";"#"&Test$&"H"
1758         GOTO Efn
1760      END IF
1762      IF POS(Test$,"*")>1 THEN
1764         Test$=Test$[1,POS(Test$,"*")-1]
1766         Maskflag=1
1768      ELSE
1770         Maskflag=0
1772      END IF
1774      FOR I=1 TO LEN(Test$)
1776         Ascii_num=NUM(Test$[I])
1778         SELECT Ascii_num
1780         CASE 65 TO 90,95,97 TO 122,48 TO 57
```

```
1782            !Allowed characters
1784          CASE ELSE
1786            BEEP
1788            DISP "ERROR in NAME ENTRY-ILLEGAL CHARACTERS, TRY AGAIN."
1790            WAIT 1.8
1792            OUTPUT 2 USING "K,#";"#"&Test$&"H"
1794            GOTO Efn
1796          END SELECT
1798        NEXT I
1800        IF Maskflag THEN
1802          Filename$=Test$&"**"
1804        ELSE
1806          Filename$=Test$
1808        END IF
1810        Ms_path$=Hfs_temp$
1812        SUBEXIT
1814 Abortline:Filename$=""
1816        IF Ac$="CAT" THEN Ms_path$=Hfs_temp$
1818        SUBEXIT
1820      SUBEND
1822      !
1824      ! ********************************************************************
1826      !
1828      SUB File_menu(Mask$,Ftype$,Fls$(*),INTEGER Fls_cnt,Dir_on,Prt_on)
1830 File_menu:  !
1832        ! Original: 29 Jun 1987, G. Koepke
1834        ! Revision: 02 Dec 1987, 07:00
1836        OPTION BASE 1
1838        DEG
1840        COM /Sys/ Sys_id$[10]
1842        COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
1844        COM /Interrupts/ INTEGER Intr_prty
1846        COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
1848        DIM Directory$(600)[80],Bd$(600)[71]
1850        DIM D$[80],T$[51],Ids$[40],Stat$[1],Test$[256]
1852        INTEGER Bd_cnt,File_cnt,I,C_cnt,C0(1),Format_error,End_search
1854        IF Fls_cnt>0 THEN ALLOCATE INTEGER Choose(Fls_cnt)
1856        !
1858        ! Catalog the disk specified
1860        !
1862        End_search=0
1864        REPEAT  ! Generate path to file and extract file name.
1866          ON ERROR GOTO Cat_errors
1868          DISP " Reading the Directory ... "
1870          IF LEN(Ms_path$)>0 THEN
1872            MASS STORAGE IS Ms_path$[1,LEN(Ms_path$)-1]&Diskdrive$
1874          ELSE
1876            MASS STORAGE IS Diskdrive$
1878          END IF
1880          CAT TO Directory$(*);NO HEADER,COUNT File_cnt
1882          OFF ERROR
1884          !
1886          ! set up array of legal file names.
1888          !
1890          Bd_cnt=0
1892          MAT Bd$= ("")
```

```
1894        FOR I=1 TO File_cnt
1896           SELECT Directory$(I)[32,36]
1898           CASE Ftype$          ! Ftype$ = "BDAT " or
1900                             ! Ftype$ = "PROG "
1902              IF LEN(Mask$)>0 THEN ! Test for mask$
1904                 IF Directory$(I)[1,LEN(Mask$)]=Mask$ THEN
1906                    Bd_cnt=Bd_cnt+1
1908                    Bd$(Bd_cnt)=Directory$(I)[1;14]&" - "&Ftype$
1910                 END IF
1912              ELSE
1914                 Bd_cnt=Bd_cnt+1
1916                 Bd$(Bd_cnt)=Directory$(I)[1;14]&" - "&Ftype$
1918              END IF
1920           CASE "DIR "          ! plus all "DIR " listings
1922              Bd_cnt=Bd_cnt+1
1924              Bd$(Bd_cnt)=Directory$(I)[1;14]&" - DIR "
1926           CASE ELSE
1928           END SELECT
1930        NEXT I
1932        IF LEN(Ms_path$)>0 AND Bd_cnt>0 AND Fls_cnt>0 THEN
1934           Bd_cnt=Bd_cnt+1
1936           Bd$(Bd_cnt)="--  MOVE back up ONE Directory level."
1938           Bd_cnt=Bd_cnt+1
1940           Bd$(Bd_cnt)="--  RETURN to ROOT Directory."
1942        END IF
1944        !
1946        ! set up file menu
1948        !
1950        D$=" Select "&VAL$(Fls_cnt)&" file name(s) for data entry."
1952        T$="List of "&Ftype$&"files and DIRs on "&Diskdrive$
1954        IF LEN(Mask$)>0 THEN
1956           T$=T$&" mask="&Mask$
1958        END IF
1960        IF Bd_cnt>0 THEN
1962           IF Dir_on>0 THEN GOSUB Read_data_id
1964           IF Prt_on THEN
1966              GOSUB List_directory
1968              End_search=1
1970           ELSE
1972              C_cnt=Fls_cnt
1974              DISP CHR$(12)
1976              IF Fls_cnt>0 THEN
1978                 CALL Menu_scroll(D$,T$,Bd$(*),Bd_cnt,C_cnt,Choose(*))
1980              ELSE
1982                 CALL Menu_scroll(D$,T$,Bd$(*),Bd_cnt,C_cnt,C0(*))
1984              END IF
1986              !
1988              ! transfer file names to Fls$(*).
1990              !
1992              IF C_cnt=0 THEN ! selection process aborted
1994                 End_search=1
1996                 MAT Fls$= ("")
1998              ELSE
2000                 MAT SORT Choose(*)
2002                 FOR I=1 TO C_cnt
2004                    IF Bd$(Choose(I))[18,22]=Ftype$ THEN
```

```
2006                      Fls$(I)=Bd$(Choose(I))[1;14]
2008                      End_search=1
2010                    ELSE  ! it must be a Directory or message.
2012                      SELECT Bd$(Choose(I))[18,22]
2014                      CASE "up ON" ! move up one directory
2016                        LOOP
2018                          Ms_path$=Ms_path$[1,LEN(Ms_path$)-1]
2020                        EXIT IF LEN(Ms_path$)=0
2022                          Test$=Ms_path$[LEN(Ms_path$);1]
2024                        EXIT IF Test$="/"
2026                        END LOOP
2028                      CASE "ROOT "  ! jump to root directory
2030                        Ms_path$=""
2032                      CASE "DIR  "  ! add directory to Ms_path$
2034                        Test$=TRIM$(Bd$(Choose(I))[1,14])
2036                        Ms_path$=Ms_path$&Test$&"/"
2038                      CASE ELSE
2040                        DISP "ERROR in directory jump"
2042                        PAUSE
2044                      END SELECT
2046                      I=C_cnt
2048                    END IF
2050                  NEXT I
2052                END IF
2054              END IF
2056            ELSE
2058              DISP " This directory contains no ";Ftype$;" files ... "
2060              WAIT 2.5
2062              End_search=1
2064            END IF
2066            DISP CHR$(12)
2068          UNTIL End_search
2070          SUBEXIT
2072 Cat_errors:!
2074          DISP " ERROR ... ";ERRM$
2076          BEEP
2078          CALL Pause_key_on
2080          DISP CHR$(12)
2082          C_cnt=0
2084          MAT Fls$= ("")
2086          SUBEXIT
2088          !
2090          ! ////////////////////////////////////////////////
2092          !
2094 Read_data_id: ! This routine expects to see Ids$ from
2096                ! GRAPH_DATA raw data files.
2098          DISP " Reading file contents ... Please stand by. "
2100          PRINT TABXY(1,18);" Reading #";
2102          FOR I=1 TO Bd_cnt    ! each BDAT file
2104            PRINT TABXY(11,18);
2106            PRINT USING "3D,4A,3D,2A,#";I," of ",Bd_cnt,". "
2108            Ids$="Data not recognized."
2110            IF Bd$(I)[18,22]="BDAT " THEN
2112              ON ERROR GOTO Not_recognized
2114              ASSIGN @Io_path TO Bd$(I)[1;14]
2116              ENTER @Io_path;Stat$
```

```
2118            SELECT Stat$
2120            CASE "N"
2122               ENTER @Io_path;Ids$
2124            CASE "Y"
2126               Ids$="Complete graph in GRAPH_DATA form."
2128            END SELECT
2130 Not_recognized:ASSIGN @Io_path TO *
2132            OFF ERROR
2134            IF Dir_on=2 THEN
2136               GOSUB Interpret_1
2138               IF Format_error THEN GOTO Other_format
2140               GOTO Go_on
2142            END IF
2144 Other_format:!
2146            Bd$(I)[23,71]=" ... "&Ids$
2148         END IF
2150 Go_on:NEXT I
2152      PRINT TABXY(1,18);RPT$(" ",40);
2154      DISP CHR$(12);
2156      RETURN
2158      !
2160      ! ////////////////////////////////////////////////
2162      !
2164 Interpret_1: ! This is used to interpret ID strings.
2166      Format_error=1
2168      ! identify this particular format
2170      RETURN
2172      !
2174      ! ////////////////////////////////////////////////
2176      !
2178 List_directory: ! This routine will provide a tabular listing of
2180               ! the directory along with Ids$ if provided
2182               !
2184      DISP " Listing directory ... "
2186      ON TIMEOUT 7,10 GOTO Printer_kaput
2188      PRINTER IS Printer
2190      PRINT USING "//"
2192      PRINT T$
2194      IF LEN(Ms_path$)>0 THEN PRINT "HFS Path:  ";Ms_path$
2196      PRINT RPT$("=",80)
2198      PRINT "File name";
2200      IF Dir_on THEN
2202         PRINT "      - TYPE  ... contents"
2204      ELSE
2206         PRINT "      - TYPE"
2208      END IF
2210      PRINT RPT$("-",80)
2212      FOR I=1 TO Bd_cnt
2214         IF Bd$(I)[18,22]=Ftype$ OR Bd$(I)[18,22]="DIR  " THEN
2216            PRINT Bd$(I)
2218         END IF
2220      NEXT I
2222      PRINT RPT$("_",80)
2224      PRINT
2226      PRINTER IS CRT
2228      OFF TIMEOUT 7
```

```
2230      RETURN
2232 Printer_kaput:DISP " Printer not responding ... listing aborted. "
2234      BEEP
2236      WAIT 1.8
2238      OFF TIMEOUT 7
2240      RETURN
2242    SUBEND
2244    !
2246    ! ************************************************************
2248    !
2250    SUB Menu_scroll(D$,T$,Items$(*),INTEGER Item_cnt,To_select,Choose(*))
2252 Menu_scroll:! Original: 22 Jun 1987, Galen Koepke, NBS 723.04
2254            ! Revision: 22 Aug 1990, 12:00, Dennis Carnell
2256            !
2258            ! A general purpose menu utility for scrolling items and
2260            ! selecting either a fixed number or a random number
2262            ! of items.
2264            !  for fixed  : To_select > 0
2266            !  for random : To_select = -1
2268            ! The items are arranged in screens of 15 items each and
2270            ! the user may access screens via softkeys. There may be
2272            ! up to 40 screens or 600 items to choose from.
2274            ! Maximum sizes: D$[80], T$[51], Items(*)[70]
2276            ! Items$(*) contains the item descriptions
2278            ! Item_cnt is the number of items in Items$(*)
2280            ! Choose(*) is dimensioned to the number of required choices
2282            !          and will be filled with the item numbers chosen.
2284            ! To_select is the number of required choices.
2286            !
2288      OPTION BASE 1
2290      PRINTER IS CRT
2292      DEG
2294      GOSUB Def_variables
2296      GOSUB Define_screens
2298      GOSUB Make_selections
2300      IF Null_file THEN  ! reset to zero
2302         Item_cnt=0
2304         Items$(1)=""
2306         To_select=0    ! no valid selections
2308      END IF
2310      SUBEXIT
2312      !
2314      ! ////////////////////////////////////////////////
2316      !
2318 Def_variables:!
2320      COM /Interrupts/ INTEGER Intr_prty
2322      COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
2324      COM /Sys/ Sys_id$[10]
2326      !
2328      INTEGER Screen_cnt,Items_per_scn,First_item(40),Last_item(40)
2330      INTEGER I,J,K,First_line,Last_line,Active_screen,Pointer,Last_pt
2332      INTEGER Local_prty,Skips,Knobcount,Pointeractive,K0,Null_file
2334      INTEGER Exit_flag,Temp,Random_select,Indx
2336      DIM Marker$[8],Test$[256]
2338      !
2340      ! initialize parameters
```

```
2342      !
2344      Local_prty=Intr_prty
2346      IF Local_prty<1 THEN Local_prty=10
2348      IF LEN(Sys_id$)=0 THEN Sys_id$=SYSTEM$("SYSTEM ID")
2350      IF Item_cnt<1 THEN
2352         Null_file=1
2354         Item_cnt=1
2356         To_select=0
2358         Items$(1)="**** Empty ****"
2360      ELSE
2362         Null_file=0
2364      END IF
2366      IF To_select=-1 THEN
2368         Random_select=1   ! choose random number of items
2370         To_select=0       ! needed for softkeys
2372      END IF
2374      IF To_select>Item_cnt THEN To_select=Item_cnt
2376      MAT Choose= (999)
2378      Skips=0
2380      Knobcount=0
2382      Doneflag=0
2384      Marker$="=====>"&RPT$(CHR$(8),4)
2386      RETURN
2388      !
2390      ! ////////////////////////////////////////////////
2392      !
2394 Define_screens:! Set up screens of 15 items each.
2396            !
2398      Items_per_scn=15    ! Maximum number of displayable items
2400      IF INT(Item_cnt/Items_per_scn)=Item_cnt/Items_per_scn  THEN
2402         Screen_cnt=INT(Item_cnt/Items_per_scn)
2404      ELSE
2406         Screen_cnt=INT(Item_cnt/Items_per_scn)+1
2408      END IF
2410      J=1
2412      FOR I=1 TO Screen_cnt   ! set up each screen
2414         First_item(I)=J
2416         IF J+Items_per_scn-1<Item_cnt THEN
2418            Last_item(I)=J+Items_per_scn-1
2420            J=J+Items_per_scn
2422         ELSE
2424            Last_item(I)=Item_cnt
2426         END IF
2428      NEXT I
2430      RETURN
2432      !
2434      ! ////////////////////////////////////////////////
2436      !
2438 Make_selections:! MENU setup and use.
2440      Active_screen=1   ! first screen is active
2442      First_line=2      ! first printed line on screen  =  2 or greater.
2444      GOSUB Write_screen  ! activate screen at Active_screen
2446                          ! and set First_line and Last_line for Pointer
2448                          ! write Marker$ to first non-selected line.
2450      K0=0                ! Keys start at zero
2452      Exit_flag=0         ! allow ENTER key to exit when selections filled.
```

```
2454 Key_loop:  !
2456      ON KBD,Local_prty GOSUB Process_kbd
2458      ON KNOB .01,Local_prty GOSUB Move_pointer
2460      IF Random_select THEN
2462         ! set keys for random selection
2464         DISP D$
2466         ON KEY K0 LABEL " Select",Local_prty GOSUB Select_random
2468         ON KEY K0+9 LABEL " Accept",Local_prty GOTO Exit_line
2470      ELSE ! set key K0 for fixed selection
2472         IF Skips<To_select THEN
2474            DISP D$
2476            IF To_select>1 THEN
2478               Test$=" Select "&VAL$(Skips+1)&" of "&VAL$(To_select)
2480            ELSE
2482               Test$=" Select"
2484            END IF
2486            ON KEY K0 LABEL Test$,Local_prty GOSUB Select_fixed
2488         ELSE
2490            IF To_select>0 THEN
2492               DISP " Selection process complete ..."
2494            ELSE
2496               DISP " Menu for information only ... "
2498            END IF
2500            ON KEY K0 LABEL "Accept",Local_prty GOTO Exit_line
2502         END IF
2504      END IF
2506      IF Active_screen<Screen_cnt THEN
2508         ON KEY K0+1 LABEL " Next  Screen",Local_prty GOSUB Next_screen
2510      ELSE
2512         OFF KEY K0+1
2514      END IF
2516      IF Active_screen>1 THEN
2518         ON KEY K0+2 LABEL " Last  Screen",Local_prty GOSUB Last_screen
2520      ELSE
2522         OFF KEY K0+2
2524      END IF
2526      IF Skips>0 OR Random_select THEN
2528         ON KEY K0+3 LABEL " Reset  Select",Local_prty GOSUB Select_reset
2530      ELSE
2532         OFF KEY K0+3
2534      END IF
2536      IF To_select>0 OR Random_select THEN
2538         ON KEY K0+4 LABEL " Abort ",Local_prty GOTO Escape_line
2540      ELSE
2542         OFF KEY K0+4
2544      END IF
2546      IF Screen_cnt>2 THEN
2548         ON KEY K0+6 LABEL "Jump to Screen",Local_prty GOSUB Jump_to_scn
2550      ELSE
2552         OFF KEY K0+6
2554      END IF
2556      IF Exit_flag THEN Exit_line
2558      GOTO Key_loop
2560 Escape_line:Skips=0
2562      MAT Choose= (0)
2564      To_select=0
```

```
2566 Exit_line:OFF KEY
2568      MAT SORT Choose(*)
2570      OFF KNOB
2572      OFF KBD
2574      OUTPUT KBD;CHR$(255)&CHR$(75);
2576      PRINT CHR$(128);
2578      ! everything cleared, now go back to work.
2580      RETURN
2582      !
2584      ! ////////////////////////////////////////////////
2586      !
2588 Next_screen:      !
2590      OFF KBD
2592      OFF KNOB
2594      OFF KEY
2596      IF Active_screen=Screen_cnt THEN RETURN
2598      Active_screen=Active_screen+1
2600      GOSUB Write_screen
2602      RETURN
2604      !
2606      ! ////////////////////////////////////////////////
2608      !
2610 Last_screen:      !
2612      OFF KBD
2614      OFF KNOB
2616      OFF KEY
2618      IF Active_screen=1 THEN RETURN
2620      Active_screen=Active_screen-1
2622      GOSUB Write_screen
2624      RETURN
2626      !
2628      ! ////////////////////////////////////////////////
2630      !
2632 Jump_to_errors:DISP " Not a valid screen number ... try again. "
2634      BEEP
2636      WAIT 1.8
2638 Jump_to_scn: !
2640      OFF KBD
2642      OFF KNOB
2644      OFF KEY
2646      DISP " ENTER the screen number desired (1 to ";Screen_cnt;").";
2648      LINPUT Test$
2650      Test$=TRIM$(Test$)
2652      IF LEN(Test$)=0 THEN Jump_to_return
2654      ON ERROR GOTO Jump_to_errors
2656      Temp=INT(VAL(Test$))
2658      OFF ERROR
2660      IF Temp<1 OR Temp>Screen_cnt THEN Jump_to_errors
2662      Active_screen=Temp
2664      GOSUB Write_screen
2666 Jump_to_return: !
2668      DISP CHR$(12)
2670      Test$=""
2672      RETURN
2674      !
2676      ! ////////////////////////////////////////////////
```

```
2678     !
2680 Select_fixed:!
2682     OFF KBD
2684     OFF KNOB
2686     OFF KEY
2688     IF NOT Pointeractive THEN
2690        DISP "NO additional selections for this screen."
2692        BEEP
2694        WAIT 2
2696        DISP CHR$(12);
2698        RETURN
2700     END IF
2702     IF Skips=To_select THEN
2704        IF To_select=0 THEN
2706           DISP "This menu is for information only,";
2708           DISP " no selection allowed."
2710        ELSE
2712           DISP "All selections have been filled,";
2714           DISP " 'Select Reset' to repeat."
2716        END IF
2718        BEEP
2720        WAIT 2
2722        DISP CHR$(12);
2724        RETURN
2726     END IF
2728     Skips=Skips+1
2730     Choose(Skips)=First_item(Active_screen)+Pointer-First_line
2732     PRINT CHR$(129);  ! inverse video
2734     PRINT TABXY(10,Pointer);Items$(Choose(Skips))
2736     PRINT CHR$(128);
2738     PRINT TABXY(1,Pointer);
2740     SELECT Pointer
2742     CASE First_line
2744        GOSUB Point_forward
2746     CASE Last_line
2748        GOSUB Point_backward
2750     CASE ELSE
2752        ! move forward unless it requires wrapping to beginning.
2754        IF Skips-1>0 THEN  ! check for selected items.
2756           I=Pointer-First_line
2758           LOOP
2760              K=0
2762              FOR J=1 TO Skips
2764                 IF First_item(Active_screen)+I=Choose(J)  THEN K=1
2766              NEXT J
2768           EXIT IF K=0
2770              I=I+1
2772              IF I+First_line>Last_line THEN K=-1
2774           EXIT IF K=-1
2776           END LOOP
2778           IF K=0 THEN
2780              GOSUB Point_forward
2782           ELSE
2784              GOSUB Point_backward
2786           END IF
2788        ELSE
```

```
2790            GOSUB Point_forward
2792          END IF
2794       END SELECT
2796       RETURN
2798       !
2800       ! ////////////////////////////////////////////////
2802       !
2804 Select_random:!
2806       OFF KBD
2808       OFF KNOB
2810       OFF KEY
2812       Test$="NO"
2814       IF NOT Pointeractive THEN
2816          DISP "NO additional selections for this screen."
2818          BEEP
2820          WAIT 2
2822          DISP CHR$(12);
2824          RETURN
2826       END IF
2828       FOR I=1 TO To_select
2830          IF Choose(I)=First_item(Active_screen)+Pointer-First_line  THEN
2832             Indx=I
2834             Test$="YES"
2836          END IF
2838       NEXT I
2840       SELECT Test$
2842       CASE "YES"              ! Selected item is tagged ... untag
2844          IF Pointer<>Last_item(Active_screen)+1  AND Pointer<>17 THEN
2846             PRINT CHR$(128);! normal video
2848          ELSE
2850             PRINT CHR$(132);! underline video
2852          END IF
2854          PRINT TABXY(10,Pointer);Items$(Choose(Indx))
2856          FOR I=Indx TO To_select-1
2858             Choose(I)=Choose(I+1)
2860          NEXT I
2862          Choose(To_select)=999
2864          To_select=To_select-1
2866       CASE "NO"               ! Selected item is untagged ... tag it
2868          To_select=To_select+1
2870          Choose(To_select)=First_item(Active_screen)+Pointer-First_line
2872          IF Pointer<>Last_item(Active_screen)+1  AND Pointer<>17 THEN
2874             PRINT CHR$(129);! inverse video
2876          ELSE
2878             PRINT CHR$(133);! inverse video with underline
2880          END IF
2882          PRINT TABXY(10,Pointer);Items$(Choose(To_select))
2884       END SELECT
2886       PRINT CHR$(128);
2888       PRINT TABXY(1,Pointer);
2890       RETURN
2892       !
2894       ! ////////////////////////////////////////////////
2896       !
2898 Select_reset:      !Clear Choose file
2900       OFF KBD
```

```
2902      OFF KNOB
2904      OFF KEY
2906      IF Random_select THEN To_select=0
2908      Skips=0
2910      MAT Choose= (999)
2912      GOSUB Write_screen
2914      RETURN
2916      !
2918      ! ////////////////////////////////////////////////
2920      !
2922 Process_kbd:! Allow use of arrows and enter key in addition to soft.
2924      Test$=KBD$
2926      IF LEN(Test$)=1 AND Test$[1,1]<>CHR$(32) THEN
2928         BEEP 80.,.1
2930         RETURN
2932      END IF
2934      IF Test$[1,1]=CHR$(32) THEN GOSUB Point_forward
2936      IF Test$[1,1]<>CHR$(255) THEN RETURN
2938      SELECT Test$[2,2]
2940      CASE CHR$(255)
2942            ! do nothing
2944      CASE "V","T"
2946         GOSUB Point_forward
2948      CASE "^","W"
2950         GOSUB Point_backward
2952      CASE "E","s","t","&"
2954         IF Random_select THEN
2956            GOSUB Select_random
2958         ELSE
2960            IF Skips<To_select THEN
2962               GOSUB Select_fixed
2964            ELSE
2966            ! exit routine
2968               Exit_flag=1
2970            END IF
2972         END IF
2974      CASE ELSE
2976         BEEP 80.,.1
2978      END SELECT
2980      Test$=""
2982      RETURN
2984      !
2986      ! ////////////////////////////////////////////////
2988      !
2990 Point_forward:Knobcount=5
2992      GOSUB Move_pointer
2994      RETURN
2996 Point_backward:Knobcount=-5
2998      GOSUB Move_pointer
3000      RETURN
3002      !
3004      ! ////////////////////////////////////////////////
3006      !
3008 Jog_pointer:! Move the selection pointer on the active screen.
3010            ! without regard to selected values
3012      IF Knobcount>0 THEN  ! Move forward
```

```
3014        Pointer=Pointer+1
3016      ELSE                ! Move backward
3018        Pointer=Pointer-1
3020      END IF
3022      IF Pointer<First_line THEN Pointer=Last_line
3024      IF Pointer>Last_line THEN Pointer=First_line
3026      RETURN
3028      !
3030      ! ////////////////////////////////////////////////////
3032      !
3034 Move_pointer:! Control pointer to avoid re-selection of items
3036      IF NOT Pointeractive THEN RETURN  ! No selections to be made.
3038      Knobcount=Knobcount+KNOBX-KNOBY
3040      IF ABS(Knobcount)<4 THEN RETURN
3042      Last_pt=Pointer
3044      GOSUB Jog_pointer
3046      IF Skips>0 THEN
3048        LOOP
3050          J=Pointer-First_line
3052          FOR I=1 TO Skips
3054            IF First_item(Active_screen)+J=Choose(I) THEN J=999
3056          NEXT I
3058          IF J=999 AND Pointer=Last_pt THEN Pointeractive=0
3060        EXIT IF Pointeractive=0
3062          IF J=999 THEN GOSUB Jog_pointer
3064        EXIT IF J<>999
3066        END LOOP
3068      END IF
3070      Knobcount=0
3072      OUTPUT KBD;CHR$(255)&CHR$(84);  ! Bring screen home
3074      IF Last_pt=Last_line THEN PRINT CHR$(132);
3076      PRINT "    ";
3078      IF Pointeractive THEN ! Pointer active
3080        IF Pointer=Last_line THEN
3082          PRINT CHR$(132);
3084        ELSE
3086          PRINT CHR$(128);
3088        END IF
3090        PRINT TABXY(1,Pointer);Marker$;CHR$(128);
3092      END IF
3094      RETURN
3096      !
3098      ! ////////////////////////////////////////////////////
3100      !
3102 Write_screen:! Write the screen pointed to by Active_screen
3104      ! home and clear screen
3106      OUTPUT KBD;CHR$(255)&CHR$(84)&CHR$(255)&CHR$(75);
3108      Knobcount=KNOBX-KNOBY   ! Clear knob and keyboard
3110      Knobcount=0
3112      Test$=KBD$
3114      Test$=""
3116      !
3118      PRINT TABXY(1,First_line-1);CHR$(132);" Item #| Screen #";
3120      PRINT USING "#,2D,4A,2D,3A";Active_screen," of ";Screen_cnt;" | "
3122      PRINT T$;RPT$(" ",51-LEN(T$));
3124      PRINT TABXY(80,First_line-1);"|";CHR$(128);
```

```
3126    J=0
3128    REPEAT
3130       IF J=Last_item(Active_screen)-First_item(Active_screen) THEN
3132          PRINT CHR$(132);
3134          PRINT TABXY(1,First_line+J);RPT$(" ",80)
3136       ELSE
3138          PRINT CHR$(128);
3140       END IF
3142       PRINT TABXY(5,First_line+J);
3144       PRINT USING "3D,A,#";First_item(Active_screen)+J,"|"
3146       IF Random_select THEN
3148          FOR I=1 TO To_select
3150             IF First_item(Active_screen)+J=Choose(I) THEN
3152                PRINT CHR$(129);
3154             END IF
3156          NEXT I
3158       ELSE
3160          IF Skips>0 THEN ! make this line inverse video
3162             FOR I=1 TO Skips
3164                IF First_item(Active_screen)+J=Choose(I) THEN
3166                   PRINT CHR$(129);
3168                END IF
3170             NEXT I
3172          END IF
3174       END IF
3176       PRINT TABXY(10,First_line+J);Items$(First_item(Active_screen)+J)
3178       PRINT TABXY(80,First_line+J);"|";
3180       J=J+1
3182    UNTIL J>=(Last_item(Active_screen)-First_item(Active_screen)+1)
3184    Last_line=Last_item(Active_screen)-First_item(Active_screen)
3186    Last_line=Last_line+First_line
3188    !
3190    ! set marker to first non-selected item.
3192    !
3194    Pointeractive=0
3196    IF To_select>0 OR Random_select THEN Pointeractive=1
3198    IF Skips>0 AND Pointeractive=1 THEN ! find first non-selected item
3200       J=0
3202       LOOP
3204          Pointer=First_line+J
3206          FOR I=1 TO Skips
3208             IF First_item(Active_screen)+J=Choose(I) THEN Pointer=0
3210          NEXT I
3212       EXIT IF Pointer<>0
3214          J=J+1
3216          IF First_line+J>Last_line THEN
3218             Pointeractive=0
3220             Pointer=First_line
3222          END IF
3224       EXIT IF Pointer<>0
3226       END LOOP
3228    ELSE
3230       Pointer=First_line
3232    END IF
3234    IF Pointeractive THEN
3236       IF Pointer=Last_line THEN
```

```
3238            PRINT CHR$(132);
3240         ELSE
3242            PRINT CHR$(128);
3244         END IF
3246         PRINT TABXY(1,Pointer);Marker$;CHR$(128);
3248      END IF
3250      RETURN
3252   SUBEND
3254   !
3256   ! ************************************************************
3258   !
3260   SUB Pause_key_on
3262 Pause_key_on: ! Make sure that CONTINUE key exists.
3264            ! Original: 02 Dec 1987
3266            ! Revision: 02 Dec 1987
3268      OPTION BASE 1
3270      COM /Sys/ Sys_id$[10]
3272      IF Sys_id$[1,4]="S300" THEN ! reset to S300 system keys
3274         CONTROL KBD,15;0
3276         CONTROL CRT,12;2
3278         LOAD KEY
3280      END IF
3282      PAUSE
3284      IF Sys_id$[1,4]="S300" THEN  ! set to S200 compatible keys
3286         OUTPUT KBD USING "K,#";"SCRATCH KEYX"
3288         CONTROL KBD,15;1
3290         CONTROL CRT,12;0
3292      END IF
3294      SUBEXIT
3296   SUBEND
3298   !
3300   ! ************************************************************
3302   !
3304   SUB Errortrap
3306 Errortrap: ! Original: 13 Nov 1984
3308          ! Revision: 02 Dec 1987
3310      ! Trap most errors here
3312      OPTION BASE 1
3314      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
3316      DIM File$[20],Test$[256],What$[20],Ac$[5]
3318      BEEP 400,.6
3320      SELECT ERRN
3322      CASE 54
3324         DISP "DUPLICATE FILE NAME:  ";Filename$;
3326         DISP "....PURGE old one? (Y/N)";
3328         LINPUT What$
3330         What$=TRIM$(What$)
3332         SELECT What$[1,1]
3334         CASE "Y","y"
3336            PURGE Ms_path$&Filename$&Diskdrive$
3338         CASE ELSE
3340            Ac$="VALID"
3342            CALL Enterfilename(Ac$)
3344         END SELECT
3346      CASE 52,53
3348         DISP "Improper FILE NAME -- ENTER NEW FILE NAME";
```

```
3350        OUTPUT 2 USING "#,K,K";"#";Filename$
3352        LINPUT Filename$
3354        Filename$=TRIM$(Filename$)
3356      CASE 56
3358        DISP "FILE:  ";Filename$;" is not on this disk, please insert";
3360        DISP " correct disk"
3362        CALL Pause_key_on
3364      CASE 64
3366        DISP "This disk is full, PLEASE insert clean disk"
3368        CALL Pause_key_on
3370      CASE 56
3372        DISP "DATA INPUT disk must be in drive!!  ";
3374        DISP "...CONTINUE when ready."
3376        CALL Pause_key_on
3378      CASE 72,73,76
3380        DISP Diskdrive$;
3382        DISP " is not available, type correct";
3384        DISP " unit specifier (ie. ':,707,0')."
3386        OUTPUT 2 USING "K,#";Diskdrive$
3388        LINPUT Diskdrive$
3390      CASE 80
3392        DISP "CHECK DISK drive door!"
3394        CALL Pause_key_on
3396      CASE ELSE
3398        DISP ERRM$;" 'CONTINUE' when fixed"
3400        CALL Pause_key_on
3402      END SELECT
3404      DISP CHR$(12)
3406      SUBEXIT
3408    SUBEND
3410    !
3412    ! ********************************************************************
3414    !
3416    SUB Auto_format(Value)
3418  Auto_format: ! Original: 13 Nov 1984
3420           ! Revision: 30 Aug 1990
3422      ! Select the proper number of digits to display.
3424      ! This routine is used by several program sections to
3426      ! print numbers to the display.
3428      !
3430      SELECT ABS(Value)
3432      CASE >=1.0E+5
3434        SELECT ABS(Value)
3436        CASE <1.0E+10
3438          PRINT USING "#,MD.4DESZ,2X";Value
3440        CASE <1.0E+100
3442          PRINT USING "#,MD.4DESZZ,X";Value
3444        CASE ELSE
3446          PRINT USING "#,MD.4DESZZZ";Value
3448        END SELECT
3450      CASE >=1.0
3452        IF Value=PROUND(Value,-4) THEN
3454          IF INT(Value)=Value THEN
3456            PRINT USING "#,M5D,6X";Value
3458          ELSE
3460            PRINT USING "#,M5D.4D,X";Value
```

```
3462          END IF
3464        ELSE
3466          PRINT USING "#,MD.4DESZ,2X";Value
3468        END IF
3470        !
3472        !
3474        !+++++++All values less that 1.0 ++++++++++++
3476        !
3478      CASE >=1.0E-4
3480        IF PROUND(Value,-4)=Value THEN
3482          PRINT USING "#,4X,MZ.4D,X";Value
3484        ELSE
3486          PRINT USING "#,MD.4DESZ,2X";Value
3488        END IF
3490        !
3492      CASE >=1.0E-9
3494        PRINT USING "#,MD.4DESZ,2X";Value
3496      CASE >=1.0E-99
3498        PRINT USING "#,MD.4DESZZ,X";Value
3500      CASE >1.0E-300
3502        PRINT USING "#,MD.3DESZZZ,X";Value
3504      CASE ELSE
3506        PRINT USING "#,4X,MZ.D,4X";Value
3508      END SELECT
3510      SUBEXIT
3512    SUBEND
3514    !
3516    !**********************************************************************
3518    !
```

```
100  ! RE-STORE "text_out:,1400"
102  !
104   COM /Sys/ Sys_id$[10]
106   COM /Sys_msi/ Msi_id$[20]
108  !
110  !*************************************************************************
112  ! This program reads in an ASCII file and produces a hard copy.
114  ! It ain't much, but it is useful.
116  ! The subroutines are modified versions of the GRAPH_DATA subs
118  ! of the same names.
120  ! Created at NIST; built by S.M. Chesnut.
122  !*************************************************************************
124  !
126 Date_line:  !
128          !************************************
130          ! April 26,1991
132          !************************************
134          !
136   OUTPUT KBD USING "K,#";"SCRATCH KEYE"        !ERASE SOFT KEYS
138   CONTROL KBD,15;0! sets the color of the soft keys
140   CONTROL KBD,2;1
142  !
144   !ntr_prty=1
146   CALL Read_data
148  !
150   PRINTER IS CRT
152   OUTPUT KBD USING "K,#";"LOAD KEYE"! restore the typing aid keys
154   CLEAR SCREEN
156   PRINT TABXY(1,5);"END of program. So long."
158   MASS STORAGE IS ":,1400"
160  !
162   END
164  !
166  !_____________________________________________________________________
168  !
170  !
172   SUB Load_disk_data(Basket_file(*),INTEGER Basketsize,Data_id$,INTEGER Flg)
174 Load_disk_data:   ! Original: 13 Nov 1984
176                  ! Revision: 02 Dec 1987
178      !This routine will enter data files from the disk
180      OPTION BASE 1
182      !
184      COM /Sys/ Sys_id$
186      COM /History/ Status$[1],Time_orgn$[8],Date_orgn$[11]
188      COM /History/ Time_chng$[8],Date_chng$[11],Description$[160]
190   !
192      COM /Labels/ Labels$(30)[60],INTEGER Lbl_count,REAL Lbl_addr(30,6)
194   !Lbl_addr: x, y, pen, size, LDIR, LORG
196   !
198      COM /Data_param/ INTEGER Datacount,Filesize,Curvecount,Roster(17,4)
200      COM /Data_param/ REAL Sym_size,Symbol$(17)[2],Curve_id$(17)[40]
202      COM /Data_param/ REAL Xmin_data,Xmax_data
204      COM /Data_param/ REAL Ymin_data,Ymax_data
206   !
208   !Roster: Curve#, Start Addr in File(*), Datacount, and PEN
```

```
210   !Symbol$(i)="" or "Y" => no symbol, connect pts
212   !Symbol$(i)="*Y" =>  * symbol, connect pts
214   !Symbol$(i)="*N" =>  * symbol, do not connect pts
216   !
218       COM /Background/ Graphtype$[12],Margins$(2)[10],Papersize$[1]
220       COM /Background/ REAL Pen_speed,INTEGER Backgnd_pen,Auto_time
222       COM /Background/ INTEGER Auto_file,REAL X_cross_y,Y_cross_x
224       COM /Background/ Xgrid_tick$[4],INTEGER Xmajor,Xminor
226       COM /Background/ Ygrid_tick$[4],INTEGER Ymajor,Yminor
228       COM /Background/ REAL Xmin_graph,Xmax_graph,Ymin_graph,Ymax_graph
230   !
232       COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
234       COM /Interrupts/ INTEGER Intr_prty
236       COM /Enlarge_file/ INTEGER Overflow
238       COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
240       COM /Data_stuff/ INTEGER Number,REAL Delta_x,REAL Strt_time
242   !
244       INTEGER R,Hold_size,Local_prty,Allocated,Fls_cnt
246       DIM Ac$[5],Tempfile$[10],Mask$[10],Ftype$[5],Fls$(1)[10]
248       REAL Dtime
250       OFF KEY
252       Local_prty=Intr_prty
254       !
256       !Select the disk drive where the data exists
258       !
260       IF Overflow<>0 THEN Overflow=0
262       Hold_size=0
264       Dtime=0.
266       Allocated=0
268 Selectdrive:      !
270       IF Diskdrive$="NO DISK" THEN Diskdrive$=""
272       IF LEN(Diskdrive$)>0 THEN GOTO Choosefilename
274       GRAPHICS OFF
276       OUTPUT 2 USING "#,K";"K"
278       CALL Select_disk
280       IF Diskdrive$="NO DISK" THEN GOTO Mistakelineset
282 Choosefilename:  !
284       Tempfile$=Filename$
286       IF LEN(Filename$)>0 THEN GOTO Bring_in_data
288       Ac$="CAT"
290       CALL Enterfilename(Ac$)
292       IF LEN(Filename$)=0 OR POS(Filename$,"*")>1 THEN
294          IF POS(Filename$,"*")>1 THEN ! set mask$
296             Mask$=Filename$[1,POS(Filename$,"*")-1]
298             Filename$=""
300          ELSE
302             Mask$=""! no preselection
304          END IF
306          Ftype$="ASCII"  ! examine ASCII files only
308          Fls_cnt=1  ! select one file
310          Intr_prty=Local_prty+1
312          CALL File_menu(Mask$,Ftype$,Fls$(*),Fls_cnt,0,0)
314          Intr_prty=Local_prty
316          Filename$=Fls$(1)
318          IF LEN(Filename$)=0 THEN  ! aborted
320             Filename$=Tempfile$
322             GOTO Mistakelineset
324          END IF
```

```
326       END IF
328 Bring_in_data:  !
330       !
332       !Find this file on the disk.
334       !
336       ON ERROR GOTO Cant_findfile
338       ASSIGN @Datapath TO Filename$&Diskdrive$
340       OFF ERROR
342       Dtime=TIMEDATE
344       DISP " LOADING disk file: ";Filename$;" ... ";
346       ON ERROR GOTO Bad_file
348       ENTER @Datapath;Status$
350       OFF ERROR
352       ON ERROR GOTO Cant_findfile
354       SELECT Status$
356       CASE "Y"    ! All graphics/data parameters exist.REN 100,2
358          DISP " Complete graph. "
360          ENTER @Datapath;Time_orgn$,Date_orgn$
362          ENTER @Datapath;Time_chng$,Date_chng$
364          ENTER @Datapath;Description$
366          ENTER @Datapath;Labels$(*),Lbl_count,Lbl_addr(*)
368          ENTER @Datapath;Curve_id$(*),Symbol$(*)
370          ENTER @Datapath;Roster(*),Curvecount
372          ENTER @Datapath;Graphtype$,Margins$(*)
374          ENTER @Datapath;X_cross_y,Y_cross_x
376          ENTER @Datapath;Xgrid_tick$,Xmajor,Xminor
378          ENTER @Datapath;Ygrid_tick$,Ymajor,Yminor
380          ENTER @Datapath;Xmin_graph,Xmax_graph
382          ENTER @Datapath;Ymin_graph,Ymax_graph
384       CASE "N"     ! Only data parameters exist.
386          DISP " RAW data. "
388       CASE ELSE
390 Bad_file:  DISP CHR$(12)
392          DISP "Data file is not recognized, entry aborted.";
394          DISP " ...continue."
396          BEEP
398          PAUSE
400          OFF ERROR
402          GOTO Mistakelineset
404       END SELECT
406       !
408       ENTER @Datapath;Data_id$
410       IF Flg THEN
412          ENTER @Datapath;Delta_x
414          ENTER @Datapath;Datacount
416          Hold_size=Datacount
418       ELSE
420          ENTER @Datapath;Datacount
422          ENTER @Datapath;Hold_size
424       END IF
426       IF NOT Allocated THEN
428          IF Datacount>=1 AND Hold_size>=1 THEN
430             ALLOCATE Holding_file(Hold_size,2)
432          ELSE
434             ALLOCATE Holding_file(1,2)
436          END IF
```

```
438         Allocated=1
440       END IF
442       ENTER @Datapath;Holding_file(*)
444       ASSIGN @Datapath TO *
446       OFF ERROR
448       IF NOT Flg THEN
450         Delta_x=Holding_file(2,1)-Holding_file(1,1)
452         Strt_time=Holding_file(1,1)
454       END IF
456       IF Datacount=0 THEN Mistakeline
458       !
460       !Copy data from Holding_file(*) to Basket_file(*)
462       !
464       MAT Basket_file= (0.)
466       IF Datacount>Basketsize THEN   !Receiving file too small.
468         Allocated=0
470         DEALLOCATE Holding_file(*)
472         DISP " DATA FILE overflow, new data discarded. ";
474         DISP " (continue) "
476         BEEP
478         PAUSE
480         IF Status$="Y" THEN
482           Curvecount=0
484           MAT Roster= (0)
486         END IF
488         Overflow=Hold_size
490         GOTO Mistakelineset
492       END IF
494 Copydatafile:      !
496       FOR R=1 TO Datacount
498         Basket_file(R,1)=Holding_file(R,1)
500         Basket_file(R,2)=Holding_file(R,2)
502       NEXT R
504       Basketsize=Datacount
506       GOTO Mistakeline
508       !
510 Mistakelineset:Datacount=0
512 Mistakeline:OFF KEY
514       IF Allocated THEN DEALLOCATE Holding_file(*)
516       LOOP
518       EXIT IF TIMEDATE-Dtime>1.8
520       END LOOP
522       DISP CHR$(12)
524       OUTPUT 2 USING "#,K";"K"
526       SUBEXIT
528       !
530       ! ////////////////////////////////////////////////////
532       !
534 Cant_findfile:      !Error in searching for the file.
536       BEEP 500,.6
538       SELECT ERRN
540       CASE 56
542         DISP "That file does not exist on this disk  ";
544       CASE 72,73,76,82
546         DISP Diskdrive$;" has failed or is not available ";
548       CASE ELSE
```

```
550         DISP ERRM$;
552       END SELECT
554       DISP " ....CONTINUE to try again."
556       PAUSE
558       Filename$ = ""
560       Diskdrive$ = ""
562       GOTO Selectdrive
564       !
566     SUBEND
568     !
570     ! ***************************************************************
572     !
574     SUB Select_disk
576   Select_disk:  ! Original: 04 Jul 1987
578             ! Revision: 06 Aug 1987
580       COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
582       COM /Interrupts/ INTEGER Intr_prty
584       COM /Sys_msi/ Msi_id$
586       COM /Sys/ Sys_id$
588       INTEGER Local_prty,Dd,Pt,Choose(1)
590       DIM Disc$(30)[60],Title$[40],Displ$[60]
592       Local_prty = Intr_prty
594       OFF KEY
596       !
598       ! Define the disk drives available for this system, reserve the
600       ! first characters for the drive address and the characters after
602       ! the - for a description of the drive.
604       !
606       ! Example:
608       ! Disc$(1) = ":,700,0,0      HP 9133H HARD disk, volume 0."
610       !
612       !
614       Displ$ = " SELECT DISK DRIVE  ... Abort will cancel. "
616       Title$ = " Available disk drives for this system. "
618       Pt = 1   ! allow only one select
620       !
622       IF Diskdrive$[1,1] < > ":" THEN Diskdrive$ = ""
624       IF Msi_id$[1,1] < > ":" THEN Msi_id$ = SYSTEM$("MSI")
626       Diskdrive$ = TRIM$(Diskdrive$)
628       Msi_id$ = TRIM$(Msi_id$)
630       IF LEN(Diskdrive$) > 0 AND LEN(Msi_id$) > 0 THEN
632         Disc$(1) = Diskdrive$&RPT$(" ",17-LEN(Diskdrive$))
634         Disc$(1) = Disc$(1)&"- Last selected disk drive."
636         Dd = 1
638         IF Diskdrive$ < > Msi_id$ THEN
640           Disc$(2) = Msi_id$&RPT$(" ",17-LEN(Msi_id$))
642           Disc$(2) = Disc$(2)&"- Start-up mass storage unit specifier."
644           Dd = Dd + 1
646         ELSE
648           Disc$(1) = Disc$(1)&"  Start-up MSUS."
650         END IF
652       ELSE
654         IF LEN(Msi_id$) > 0 THEN
656           Disc$(1) = Msi_id$&RPT$(" ",17-LEN(Msi_id$))
658           Disc$(1) = Disc$(1)&"- Start-up mass storage unit specifier."
660           Dd = 1
```

```
662         ELSE
664            Dd=0
666         END IF
668      END IF
670 Disk:  !
672      ! ..... customize system drives here .....
674      ! Follow format with - after unit specifier, description is
676      ! optional but recommended.
678      ! ..........................................................
680      !
682      Disc$(Dd+1)=":,702,0          - HP 9122 dual microfloppy left drive"
684      Disc$(Dd+2)=":,702,1          - HP 9122 dual microfloppy right drive"
686      Disc$(Dd+3)=":,703,0          - HP 9125 single 5.25 floppy drive"
688      Disc$(Dd+4)=":,1400           - HP 7957B HARD DISK HFS FILE"
690      !
692      Dd=Dd+4    ! add the number of drive specifiers above
694      !
696      IF Sys_id$[1,4]<>"S300" THEN
698         Disc$(Dd+1)=":,4,1            - LEFT internal series 200"
700         Disc$(Dd+2)=":,4,0            - RIGHT internal series 200"
702         Dd=Dd+2
704      END IF
706      ! ..........................................................
708      !
710      CALL Menu_scroll(Displ$,Title$,Disc$(*),Dd,Pt,Choose(*))
712      IF Pt=0 THEN
714         Diskdrive$="NO DISK"
716      ELSE
718         Dd=POS(Disc$(Choose(Pt)),"-")-1  ! find -
720         IF Dd>5 THEN  !  valid msus
722            Diskdrive$=TRIM$(Disc$(Choose(Pt))[1,Dd])
724         ELSE
726            DISP " ERROR in reading MSUS from string, - chr not found. "
728            BEEP
730            PAUSE
732            Diskdrive$="NO DISK"
734         END IF
736      END IF
738 Diskselected:OFF KEY
740      SUBEXIT
742   SUBEND
744   !
746   ! ********************************************************
748   !
750   SUB Data_to_disk_r(REAL File(*),INTEGER Filesize,Datacount,Data_id$)
752 Data_to_disk_r:   ! Original: 13 Nov 1984
754               ! Revision: 06 Aug 1987
756      ! This routine will SAVE data files on the disk in RAW data format.
758      ! Special features:
760      !    If the Diskdrive$ and/or the Filename$ are null this routine
762      !    will prompt the operator for information.  However, if they
764      !    are not null it is assumed that the program is supplying the
766      !    correct information.
768      !
770      OPTION BASE 1
772      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
```

```
774       COM /Interrupts/ INTEGER Intr_prty
776       INTEGER Local_prty,Diskspace
778       DIM Ac$[5],Status$[1]
780       REAL Dtime
782       OFF KEY
784       Local_prty=Intr_prty
786       Dtime=0.
788       !
790       !Select the disk drive for data storage
792       !
794 Selectdrive:  !
796       IF Diskdrive$="NO DISK" THEN Diskdrive$=""
798       IF LEN(Diskdrive$)>0 THEN GOTO Choosefilename
800       GRAPHICS OFF
802       OUTPUT 2 USING "#,K";"K"
804       CALL Select_disk
806       IF Diskdrive$="NO DISK" THEN GOTO Mistakeline
808 Choosefilename:  !
810       IF LEN(Filename$)>0 THEN GOTO Send_to_disk
812       Ac$="ABORT"
814       CALL Enterfilename(Ac$)
816       IF LEN(Filename$)=0 THEN GOTO Mistakeline
818 Send_to_disk:  ! Create file and save information.
820       ON ERROR GOTO Cant_savedata
822       Diskspace=INT((Filesize*16.0)/256)+2
824       CREATE ASCII Filename$&Diskdrive$,Diskspace
826       Dtime=TIMEDATE
828       DISP " SAVING data in file ";Filename$;" on ";Diskdrive$
830       Status$="N"
832       ASSIGN @Datapath TO Filename$&Diskdrive$
834       OUTPUT @Datapath;Status$
836       OUTPUT @Datapath;Data_id$    !40 chrs description of data
838       OUTPUT @Datapath;Datacount   !number of xy points
840       OUTPUT @Datapath;Filesize    !size of array
842       OUTPUT @Datapath;File(*)
844       ASSIGN @Datapath TO *
846       OFF ERROR
848       !
850 Mistakeline:OFF KEY
852       LOOP
854       EXIT IF TIMEDATE-Dtime>1.8
856       END LOOP
858       DISP CHR$(12)
860       OUTPUT 2 USING "#,K";"K"
862       SUBEXIT
864       !
866       ! ////////////////////////////////////////////////////
868       !
870 Cant_savedata:  !
872       BEEP 500,.6
874       SELECT ERRN
876       CASE 72,73,76,78,81,82,90,93
878          DISP Diskdrive$;" has failed or is not available ";
880          DISP " ....CONTINUE to try again."
882          PAUSE
884          Diskdrive$=""
```

```
886       CASE 84,85
888          DISP " This disk is not initialized ";
890          DISP " ....CONTINUE to try again."
892          PAUSE
894          Diskdrive$ =""
896       CASE 55,64
898          DISP " This disk is full, insert new floppy and/or";
900          DISP " select new drive ...CONTINUE "
902          PAUSE
904          Diskdrive$ =""
906       CASE ELSE
908          CALL Errortrap
910          IF LEN(Filename$)>0 THEN GOTO Send_to_disk
912       END SELECT
914       GOTO Selectdrive
916       !
918    SUBEND
920    !
922    ! ***************************************************************
924    !
926    SUB Menu_scroll(D$,T$,Items$(*),INTEGER Item_cnt,To_select,Choose(*))
928 Menu_scroll:! Original: 22 Jun 1987, Galen Koepke, NBS 723.04
930           ! Revision: 06 Aug 1987, 10:00
932           !
934           ! A general purpose menu utility for scrolling items and
936           ! selecting a given number of them.
938           ! The items are arranged in screens of 15 items each and
940           ! the user may access screens via softkeys. There may be
942           ! up to 10 screens or 150 items to choose from.
944           ! Items$(*) contains the item descriptions
946           ! Item_cnt is the number of items in Items$(*)
948           ! Choose(*) is dimensioned to the number of required choices
950           !         and will be filled with the item numbers chosen.
952           ! To_select is the number of required choices.
954           !
956       OPTION BASE 1
958       PRINTER IS CRT
960       DEG
962       GOSUB Def_variables
964       GOSUB Define_screens
966       GOSUB Make_selections
968       IF Null_file THEN  ! reset to zero
970          Item_cnt=0
972          Items$(1)=""
974          To_select=0    ! no valid selections
976       END IF
978       SUBEXIT
980       !
982       ! ////////////////////////////////////////////////
984       !
986 Def_variables:!
988       COM /Interrupts/ INTEGER Intr_prty
990       COM /Bugs/ INTEGER Bug1,Bug2,Bug3,Printer
992       COM /Sys/ Sys_id$[10]
994       !
996       INTEGER Screen_cnt,Items_per_scn,First_item(10),Last_item(10)
```

```
998       INTEGER I,J,K,First_line,Last_line,Active_screen,Pointer,Last_pt
1000      INTEGER Local_prty,Skips,Knobcount,Pointeractive,K0,Null_file
1002      INTEGER Exit_flag
1004      DIM Marker$[8],Test$[160]
1006      !
1008      ! initialize parameters
1010      !
1012      Local_prty=Intr_prty
1014      IF Local_prty<1 THEN Local_prty=10
1016      IF LEN(Sys_id$)=0 THEN Sys_id$=SYSTEM$("SYSTEM ID")
1018      IF Item_cnt<1 THEN
1020         Null_file=1
1022         Item_cnt=1
1024         To_select=0
1026         Items$(1)="**** Empty ****"
1028      ELSE
1030         Null_file=0
1032      END IF
1034      IF To_select>Item_cnt THEN To_select=Item_cnt
1036      Skips=0
1038      Knobcount=0
1040      Doneflag=0
1042      Marker$="=====>"&RPT$(CHR$(8),4)
1044      RETURN
1046      !
1048      ! ////////////////////////////////////////////////
1050      !
1052 Define_screens:! Set up screens of 15 items each.
1054            !
1056      Items_per_scn=15    ! Maximum number of displayable items
1058      IF INT(Item_cnt/Items_per_scn)=Item_cnt/Items_per_scn  THEN
1060         Screen_cnt=INT(Item_cnt/Items_per_scn)
1062      ELSE
1064         Screen_cnt=INT(Item_cnt/Items_per_scn)+1
1066      END IF
1068      J=1
1070      FOR I=1 TO Screen_cnt   ! set up each screen
1072         First_item(I)=J
1074         IF J+Items_per_scn-1<Item_cnt THEN
1076            Last_item(I)=J+Items_per_scn-1
1078            J=J+Items_per_scn
1080         ELSE
1082            Last_item(I)=Item_cnt
1084         END IF
1086      NEXT I
1088      RETURN
1090      !
1092      ! ////////////////////////////////////////////////
1094      !
1096 Make_selections:! MENU setup and use.
1098      Active_screen=1   ! first screen is active
1100      First_line=2      ! first printed line on screen = 2 or greater.
1102      GOSUB Write_screen  ! activate screen at Active_screen
1104                          ! and set First_line and Last_line for Pointer
1106                          ! write Marker$ to first non-selected line.
1108      K0=0                ! Keys start at zero
```

```
1110      Exit_flag=0          ! allow ENTER key to exit when selections filled.
1112      IF Sys_id$[1,4]="S300" THEN
1114         CONTROL KBD,2;1
1116         STATUS KBD,14;J
1118         IF J=0 THEN       ! key 1 defined
1120            K0=1
1122         ELSE              ! key 0 defined
1124            K0=0
1126         END IF
1128      ELSE
1130         K0=0
1132      END IF
1134 Key_loop:  !
1136      ON KBD,Local_prty GOSUB Process_kbd
1138      ON KNOB .01,Local_prty GOSUB Move_pointer
1140      IF Skips<To_select THEN
1142         DISP D$
1144         IF To_select>1 THEN
1146            Test$=" Select  "&VAL$(Skips+1)&" of "&VAL$(To_select)
1148         ELSE
1150            Test$=" Select"
1152         END IF
1154         ON KEY K0 LABEL Test$,Local_prty GOSUB Select_item
1156      ELSE
1158         IF To_select>0 THEN
1160            DISP " Selection process complete ..."
1162         ELSE
1164            DISP " Menu for information only ... "
1166         END IF
1168         ON KEY K0 LABEL "Accept",Local_prty GOTO Exit_line
1170      END IF
1172      IF Active_screen<Screen_cnt THEN
1174         ON KEY K0+1 LABEL " Next   Screen",Local_prty GOSUB Next_screen
1176      ELSE
1178         OFF KEY K0+1
1180      END IF
1182      IF Active_screen>1 THEN
1184         ON KEY K0+2 LABEL " Last   Screen",Local_prty GOSUB Last_screen
1186      ELSE
1188         OFF KEY K0+2
1190      END IF
1192      IF Skips>0 THEN
1194         ON KEY K0+3 LABEL " Reset  Select",Local_prty GOSUB Select_reset
1196      ELSE
1198         OFF KEY K0+3
1200      END IF
1202      IF To_select>0 THEN
1204         ON KEY K0+4 LABEL " Abort ",Local_prty GOTO Escape_line
1206      ELSE
1208         OFF KEY K0+4
1210      END IF
1212      IF Exit_flag THEN Exit_line
1214      GOTO Key_loop
1216 Escape_line:Skips=0
1218      MAT Choose= (0)
1220      To_select=0
```

```
1222 Exit_line:OFF KEY
1224      OFF KNOB
1226      OFF KBD
1228      OUTPUT KBD;CHR$(255)&CHR$(75);
1230      PRINT CHR$(128);
1232      ! everything cleared, now go back to work.
1234      RETURN
1236      !
1238      ! ////////////////////////////////////////////////
1240      !
1242 Next_screen:      !
1244      OFF KBD
1246      OFF KNOB
1248      OFF KEY
1250      IF Active_screen=Screen_cnt THEN RETURN
1252      Active_screen=Active_screen+1
1254      GOSUB Write_screen
1256      RETURN
1258      !
1260      ! ////////////////////////////////////////////////
1262      !
1264 Last_screen:      !
1266      OFF KBD
1268      OFF KNOB
1270      OFF KEY
1272      IF Active_screen=1 THEN RETURN
1274      Active_screen=Active_screen-1
1276      GOSUB Write_screen
1278      RETURN
1280      !
1282      ! ////////////////////////////////////////////////
1284      !
1286 Select_item:!
1288      OFF KBD
1290      OFF KNOB
1292      OFF KEY
1294      IF NOT Pointeractive THEN
1296         DISP "NO additional selections for this screen."
1298         BEEP
1300         WAIT 2
1302         DISP CHR$(12);
1304         RETURN
1306      END IF
1308      IF Skips=To_select THEN
1310         IF To_select=0 THEN
1312            DISP "This menu is for information only,";
1314            DISP " no selection allowed."
1316         ELSE
1318            DISP "All selections have been filled,";
1320            DISP " 'Select Reset' to repeat."
1322         END IF
1324         BEEP
1326         WAIT 2
1328         DISP CHR$(12);
1330         RETURN
1332      END IF
```

```
1334      Skips = Skips + 1
1336      Choose(Skips) = First_item(Active_screen) + Pointer-First_line
1338      PRINT CHR$(129);  ! inverse video
1340      PRINT TABXY(10,Pointer);Items$(Choose(Skips))
1342      PRINT CHR$(128);
1344      PRINT TABXY(1,Pointer);
1346      SELECT Pointer
1348      CASE First_line
1350         GOSUB Point_forward
1352      CASE Last_line
1354         GOSUB Point_backward
1356      CASE ELSE
1358         ! move forward unless it requires wrapping to beginning.
1360         IF Skips-1 >0 THEN  ! check for selected items.
1362            I=Pointer-First_line
1364            LOOP
1366               K=0
1368               FOR J=1 TO Skips
1370                  IF First_item(Active_screen)+I=Choose(J)  THEN K=1
1372               NEXT J
1374            EXIT IF K=0
1376               I=I+1
1378               IF I+First_line>Last_line THEN K=-1
1380            EXIT IF K=-1
1382            END LOOP
1384            IF K=0 THEN
1386               GOSUB Point_forward
1388            ELSE
1390               GOSUB Point_backward
1392            END IF
1394         ELSE
1396            GOSUB Point_forward
1398         END IF
1400      END SELECT
1402      RETURN
1404      !
1406      ! ////////////////////////////////////////////////
1408      !
1410 Select_reset:      !Clear Choose file
1412      OFF KBD
1414      OFF KNOB
1416      OFF KEY
1418      Skips=0
1420      MAT Choose= (0)
1422      GOSUB Write_screen
1424      RETURN
1426      !
1428      ! ////////////////////////////////////////////////
1430      !
1432 Process_kbd:! Allow use of arrows and enter key in addition to soft.
1434      Test$=KBD$
1436      IF LEN(Test$)=1 AND Test$[1,1]<>CHR$(32) THEN
1438         BEEP 80,,.1
1440         RETURN
1442      END IF
1444      IF Test$[1,1]=CHR$(32) THEN GOSUB Point_forward
```

```
1446     IF Test$[1,1]<>CHR$(255) THEN RETURN
1448     SELECT Test$[2,2]
1450     CASE CHR$(255)
1452           ! do nothing
1454     CASE "V","T"
1456        GOSUB Point_forward
1458     CASE "^","W"
1460        GOSUB Point_backward
1462     CASE "E"
1464        IF Skips<To_select THEN
1466           GOSUB Select_item
1468        ELSE
1470           ! exit routine
1472           Exit_flag=1
1474        END IF
1476     CASE ELSE
1478        BEEP 80.,.1
1480     END SELECT
1482     Test$=""
1484     RETURN
1486     !
1488     ! ////////////////////////////////////////////////
1490     !
1492 Point_forward:Knobcount=5
1494     GOSUB Move_pointer
1496     RETURN
1498 Point_backward:Knobcount=-5
1500     GOSUB Move_pointer
1502     RETURN
1504     !
1506     ! ////////////////////////////////////////////////
1508     !
1510 Jog_pointer:! Move the selection pointer on the active screen.
1512           ! without regard to selected values
1514     IF Knobcount>0 THEN  ! Move forward
1516        Pointer=Pointer+1
1518     ELSE                 ! Move backward
1520        Pointer=Pointer-1
1522     END IF
1524     IF Pointer<First_line THEN Pointer=Last_line
1526     IF Pointer>Last_line THEN Pointer=First_line
1528     RETURN
1530     !
1532     ! ////////////////////////////////////////////////
1534     !
1536 Move_pointer:! Control pointer to avoid re-selection of items
1538     IF NOT Pointeractive THEN RETURN  ! No selections to be made.
1540     Knobcount=Knobcount+KNOBX+KNOBY
1542     IF ABS(Knobcount)<4 THEN RETURN
1544     Last_pt=Pointer
1546     GOSUB Jog_pointer
1548     IF Skips>0 THEN
1550        LOOP
1552           J=Pointer-First_line
1554           FOR I=1 TO Skips
1556              IF First_item(Active_screen)+J=Choose(I) THEN J=999
```

```
1558          NEXT I
1560          IF J=999 AND Pointer=Last_pt THEN Pointeractive=0
1562       EXIT IF Pointeractive=0
1564          IF J=999 THEN GOSUB Jog_pointer
1566       EXIT IF J<>999
1568       END LOOP
1570    END IF
1572    Knobcount=0
1574    OUTPUT KBD;CHR$(255)&CHR$(84);  ! Bring screen home
1576    IF Last_pt=Last_line THEN PRINT CHR$(132);
1578    PRINT "    ";
1580    IF Pointeractive THEN ! Pointer active
1582       IF Pointer=Last_line THEN
1584          PRINT CHR$(132);
1586       ELSE
1588          PRINT CHR$(128);
1590       END IF
1592       PRINT TABXY(1,Pointer);Marker$;CHR$(128);
1594    END IF
1596    RETURN
1598    !
1600    ! ////////////////////////////////////////////////
1602    !
1604 Write_screen:! Write the screen pointed to by Active_screen
1606    ! home and clear screen
1608    OUTPUT KBD;CHR$(255)&CHR$(84)&CHR$(255)&CHR$(75);
1610    Knobcount=KNOBX+KNOBY   ! Clear knob and keyboard
1612    Knobcount=0
1614    Test$=KBD$
1616    Test$=""
1618    !
1620    PRINT TABXY(1,First_line-1);CHR$(132);" Item #| Screen #";
1622    PRINT USING "#,2D,3A,2D,3A";Active_screen;" of ";Screen_cnt;" | "
1624    PRINT T$;RPT$(" ",52-LEN(T$));CHR$(128);
1626    J=0
1628    REPEAT
1630       IF J=Last_item(Active_screen)-First_item(Active_screen) THEN
1632          PRINT CHR$(132);
1634          PRINT TABXY(1,First_line+J);RPT$(" ",80)
1636       ELSE
1638          PRINT CHR$(128);
1640       END IF
1642       PRINT TABXY(5,First_line+J);
1644       PRINT USING "3D,A,#";First_item(Active_screen)+J,"|"
1646       IF Skips>0 THEN  ! make this line inverse video
1648          FOR I=1 TO Skips
1650             IF First_item(Active_screen)+J=Choose(I) THEN
1652                PRINT CHR$(129);
1654             END IF
1656          NEXT I
1658       END IF
1660       PRINT TABXY(10,First_line+J);Items$(First_item(Active_screen)+J)
1662       J=J+1
1664    UNTIL J>=(Last_item(Active_screen)-First_item(Active_screen)+1)
1666    Last_line=Last_item(Active_screen)-First_item(Active_screen)
1668    Last_line=Last_line+First_line
```

```
1670     !
1672     ! set marker to first non-selected item.
1674     !
1676     Pointeractive=0
1678     IF To_select>0 THEN Pointeractive=1
1680     IF Skips>0 AND Pointeractive=1 THEN ! find first non-selected item
1682        J=0
1684        LOOP
1686           Pointer=First_line+J
1688           FOR I=1 TO Skips
1690              IF First_item(Active_screen)+J=Choose(I) THEN Pointer=0
1692           NEXT I
1694        EXIT IF Pointer<>0
1696           J=J+1
1698           IF First_line+J>Last_line THEN
1700              Pointeractive=0
1702              Pointer=First_line
1704           END IF
1706        EXIT IF Pointer<>0
1708        END LOOP
1710     ELSE
1712        Pointer=First_line
1714     END IF
1716     IF Pointeractive THEN
1718        IF Pointer=Last_line THEN
1720           PRINT CHR$(132);
1722        ELSE
1724           PRINT CHR$(128);
1726        END IF
1728        PRINT TABXY(1,Pointer);Marker$;CHR$(128);
1730     END IF
1732     RETURN
1734  SUBEND
1736  !
1738  ! ************************************************************
1740  !
1742  SUB Errortrap
1744 Errortrap: ! Original: 13 Nov 1984
1746          ! Revision: 06 Aug 1987
1748          ! Trap most disk errors here
1750     COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
1752     DIM File$[20],Test$[160],What$[20],Ac$[5]
1754     BEEP 400,.6
1756     SELECT ERRN
1758     CASE 54
1760        DISP "DUPLICATE FILE NAME:  ";Filename$;
1762        DISP "....PURGE old one? (Y/N)";
1764        LINPUT What$
1766        What$=TRIM$(What$)
1768        SELECT What$[1,1]
1770        CASE "Y","y"
1772           PURGE Filename$&Diskdrive$
1774        CASE ELSE
1776           Ac$="VALID"
1778           CALL Enterfilename(Ac$)
1780        END SELECT
```

```
1782      CASE 52,53
1784        DISP "Improper FILE NAME -- ENTER NEW FILE NAME";
1786        OUTPUT 2 USING "#,K,K";"#";Filename$
1788        LINPUT Filename$
1790        Filename$ =TRIM$(Filename$)
1792      CASE 56
1794        DISP "FILE:  ";Filename$;" is not on this disk, please insert";
1796        DISP " correct disk"
1798        PAUSE
1800      CASE 64
1802        DISP "This disk is full, PLEASE insert clean disk"
1804        PAUSE
1806      CASE 56
1808        DISP "DATA INPUT disk must be in drive!!  ";
1810        DISP "...CONTINUE when ready."
1812        PAUSE
1814      CASE 72,73,76
1816        DISP Diskdrive$;
1818        DISP " is not available, type correct";
1820        DISP " unit specifier (ie. ':,707,0')."
1822        OUTPUT 2 USING "K,#";Diskdrive$
1824        LINPUT Diskdrive$
1826      CASE 80
1828        DISP "CHECK DISK drive door!"
1830        PAUSE
1832      CASE ELSE
1834        DISP ERRM$;" 'CONTINUE' when fixed"
1836        PAUSE
1838      END SELECT
1840      DISP CHR$(12)
1842      SUBEXIT
1844    SUBEND
1846    !
1848    ! ************************************************************************
1850    !
1852    SUB File_menu(Mask$,Ftype$,Fls$(*),INTEGER Fls_cnt,Dir_on,Prt_on)
1854  File_menu:  !
1856      ! Original: 29 Jun 1987, G. Koepke
1858      ! Revision: 06 Aug 1987, 10:00
1860      OPTION BASE 1
1862      DEG
1864      COM /Sys/ Sys_id$[10]
1866      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
1868      COM /Interrupts/ INTEGER Intr_prty
1870      DIM Directory$(150)[80],Bd$(150)[71]
1872      DIM D$[80],T$[52],Ids$[40],Stat$[1]
1874      INTEGER Bd_cnt,File_cnt,I,C_cnt,C0(1),Format_error
1876      IF Fls_cnt>0 THEN ALLOCATE INTEGER Choose(Fls_cnt)
1878      !
1880      ! Catalog the disk specified
1882      !
1884      ON ERROR GOTO Cat_errors
1886      DISP " Reading the Directory ... "
1888      MASS STORAGE IS Diskdrive$
1890      CAT TO Directory$(*);NO HEADER,COUNT File_cnt
1892      OFF ERROR
```

```
1894    !
1896    ! set up array of legal file names.
1898    !
1900    Bd_cnt=0
1902    FOR I=1 TO File_cnt
1904       IF Directory$(I)[32,36]=Ftype$ THEN  ! Ftype$="ASCII "
1906                                            ! Ftype$="PROG "
1908          IF LEN(Mask$)>0 THEN ! Test for mask$
1910             IF Directory$(I)[1,LEN(Mask$)]=Mask$ THEN
1912                Bd_cnt=Bd_cnt+1
1914                Bd$(Bd_cnt)=Directory$(I)[1;10]
1916             END IF
1918          ELSE
1920             Bd_cnt=Bd_cnt+1
1922             Bd$(Bd_cnt)=Directory$(I)[1;10]
1924          END IF
1926       END IF
1928    NEXT I
1930    !
1932    ! set up file menu
1934    !
1936    D$="Select "&VAL$(Fls_cnt)&" file names for data entry."
1938    T$="List of "&Ftype$&"files on "&Diskdrive$
1940    IF LEN(Mask$)>0 THEN
1942       T$=T$&" mask="&Mask$
1944    END IF
1946    IF Bd_cnt>0 THEN
1948       IF Dir_on>0 THEN GOSUB Read_data_id
1950       IF Prt_on THEN
1952          GOSUB List_directory
1954       ELSE
1956          C_cnt=Fls_cnt
1958          DISP CHR$(12)
1960          IF Fls_cnt>0 THEN
1962             CALL Menu_scroll(D$,T$,Bd$(*),Bd_cnt,C_cnt,Choose(*))
1964          ELSE
1966             CALL Menu_scroll(D$,T$,Bd$(*),Bd_cnt,C_cnt,C0(*))
1968          END IF
1970          !
1972          ! transfer file names to Fls$(*).
1974          !
1976          IF C_cnt=0 THEN ! selection process aborted
1978             MAT Fls$= ("")
1980          ELSE
1982             MAT SORT Choose(*)
1984             FOR I=1 TO C_cnt
1986                Fls$(I)=Bd$(Choose(I))[1;10]
1988             NEXT I
1990          END IF
1992       END IF
1994    ELSE
1996       DISP " This directory contains no ASCII files ... "
1998       WAIT 2.5
2000    END IF
2002    DISP CHR$(12)
2004    SUBEXIT
```

```
2006 Cat_errors:!
2008      DISP "ERROR ... ";ERRM$
2010      BEEP
2012      PAUSE
2014      C_cnt=0
2016      MAT Fls$= ("")
2018      SUBEXIT
2020      !
2022      ! ////////////////////////////////////////////////
2024      !
2026 Read_data_id: ! This routine expects to see lds$ from
2028            ! GRAPH_DATA raw data files.
2030      DISP " Reading file contents ... "
2032      FOR I=1 TO Bd_cnt    ! each ASCII file
2034         lds$="Data not recognized."
2036         ON ERROR GOTO Not_recognized
2038         ASSIGN @Io_path TO Bd$(I)[1;10]
2040         ENTER @Io_path;Stat$
2042         SELECT Stat$
2044         CASE "N"
2046            ENTER @Io_path;lds$
2048         CASE "Y"
2050            lds$="Complete graph ... use GRAPH_DATA."
2052         END SELECT
2054 Not_recognized:ASSIGN @Io_path TO *
2056         OFF ERROR
2058         IF Dir_on=2 THEN
2060            GOSUB Interpret_1
2062            IF Format_error THEN GOTO Other_format
2064            GOTO Go_on
2066         END IF
2068 Other_format:!
2070         Bd$(I)[11,71]=" ... "&lds$
2072 Go_on:NEXT I
2074      RETURN
2076      !
2078      ! ////////////////////////////////////////////////
2080      !
2082 Interpret_1:! This is used to interpret TEM program ID strings.
2084      Format_error=0
2086      ! identify this particular format
2088      IF LEN(lds$)<40 THEN
2090         Format_error=1
2092         RETURN
2094      END IF
2096      IF lds$[40]<>"**" THEN
2098         Format_error=1
2100         RETURN
2102      END IF
2104      ! make the information readable
2106      Bd$(I)[11,15]=" ... "
2108      Bd$(I)[16,25]=lds$[1,10]
2110      Bd$(I)[26,32]=", "&lds$[11,12]&":"&lds$[13,14]
2112      Bd$(I)[33,42]=", "&lds$[15,16]&" "&lds$[17,18]&" "&lds$[19,20]
2114      Bd$(I)[43,55]=", "&lds$[21,27]&" MHz"
2116      Bd$(I)[56,65]=", "&lds$[28,33]&"vm"
```

```
2118      Bd$(I)[66,71]="," &Ids$[38,39]
2120      RETURN
2122      !
2124      ! ////////////////////////////////////////////////
2126      !
2128 List_directory: ! This routine will provide a tabular listing of
2130              ! the directory along with Ids$ if provided
2132              !
2134      DISP " Listing directory ... "
2136      PRINTER IS PRT
2138      PRINT USING "//"
2140      PRINT T$
2142      PRINT RPT$("-",80)
2144      PRINT "File name";
2146      IF Dir_on THEN
2148         PRINT "  ...  contents"
2150      ELSE
2152         PRINT
2154      END IF
2156      PRINT RPT$("-",80)
2158      FOR I=1 TO Bd_cnt
2160         PRINT Bd$(I)
2162      NEXT I
2164      PRINT RPT$("_",80)
2166      PRINT
2168      PRINTER IS CRT
2170      RETURN
2172   SUBEND
2174   !
2176   ! ************************************************************
2178   !
2180   SUB Read_data
2182   !
2184 Read_data:  !
2186   !
2188      OPTION BASE 1
2190      RAD
2192   !
2194      COM /Sys/ Sys_id$[10]
2196      COM /Sys_msi/ Msi_id$[20]
2198      COM /Interrupts/ INTEGER Intr_prty
2200      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
2202      !
2204      INTEGER Local_prty,Diskspace,Fls_cnt
2206      DIM Ac$[5],Tempfile$[10],Mask$[10],Ftype$[5],Fls$(1)[10]
2208      DIM Data_id$[40]
2210      REAL Dtime
2212      OFF KEY
2214      Local_prty=Intr_prty
2216      Dtime=0.
2218      Filename$=""
2220      Diskdrive$=""
2222      !
2224      !
2226      !Select the disk drive where the data exists
2228      !
```

```
2230      IF Overflow<>0 THEN Overflow=0
2232      Hold_size=0
2234      Dtime=0.
2236      Allocated=0
2238 Selectdrive:      !
2240      IF Diskdrive$="NO DISK" THEN Diskdrive$=""
2242      IF LEN(Diskdrive$)>0 THEN GOTO Choosefilename
2244      GRAPHICS OFF
2246      OUTPUT 2 USING "#,K";"K"
2248      CALL Select_disk
2250      IF Diskdrive$="NO DISK" THEN GOTO Mistakelineset
2252 Choosefilename: !
2254      Tempfile$=Filename$
2256      IF LEN(Filename$)>0 THEN GOTO Bring_in_data
2258      Ac$="CAT"
2260      CALL Enterfilename(Ac$)
2262      IF LEN(Filename$)=0 OR POS(Filename$,"*")>1 THEN
2264         IF POS(Filename$,"*")>1 THEN ! set mask$
2266            Mask$=Filename$[1,POS(Filename$,"*")-1]
2268            Filename$=""
2270         ELSE
2272            Mask$=""! no preselection
2274         END IF
2276         Ftype$="ASCII"  ! examine ASCII files only
2278         Fls_cnt=1  ! select one file
2280         Intr_prty=Local_prty+1
2282         CALL File_menu(Mask$,Ftype$,Fls$(*),Fls_cnt,0,0)
2284         Intr_prty=Local_prty
2286         Filename$=Fls$(1)
2288         IF LEN(Filename$)=0 THEN  ! aborted
2290            Filename$=Tempfile$
2292            GOTO Mistakelineset
2294         END IF
2296      END IF
2298 Bring_in_data: !
2300      !
2302      !Find this file on the disk.
2304      !
2306      ON ERROR GOTO Cant_findfile
2308      ASSIGN @Datapath TO Filename$&Diskdrive$
2310      OFF ERROR
2312      Dtime=TIMEDATE
2314      DISP " LOADING disk file: ";Filename$;" ... ";
2316      ENTER @Datapath;Data_id$
2318      !
2320      ENTER @Datapath;Datacount
2322      IF NOT Allocated THEN
2324         IF Datacount>=1 THEN
2326            ALLOCATE Holding_file$(Datacount)[80]
2328         ELSE
2330            ALLOCATE Holding_file$(1)[80]
2332         END IF
2334         Allocated=1
2336      END IF
2338      ENTER @Datapath;Holding_file$(*)
2340      ASSIGN @Datapath TO *
```

```
2342      OFF ERROR
2344      IF Datacount=0 THEN Mistakeline
2346      PRINTER IS PRT
2348      FOR I=1 TO Datacount
2350         PRINT Holding_file$(I)
2352      NEXT I
2354      PRINTER IS CRT
2356      !
2358      Overflow=Hold_size
2360      GOTO Mistakeline
2362      !
2364 Mistakelineset:Datacount=0
2366 Mistakeline:OFF KEY
2368!     IF Allocated THEN DEALLOCATE Holding_file$(*)
2370      LOOP
2372      EXIT IF TIMEDATE-Dtime>1.8
2374      END LOOP
2376      DISP CHR$(12)
2378      OUTPUT 2 USING "#,K";"K"
2380      SUBEXIT
2382      !
2384      ! ////////////////////////////////////////////////////
2386      !
2388 Cant_findfile:      !Error in searching for the file.
2390      BEEP 500,.6
2392      SELECT ERRN
2394      CASE 56
2396         DISP "That file does not exist on this disk  ";
2398      CASE 72,73,76,82
2400         DISP Diskdrive$;" has failed or is not available ";
2402      CASE ELSE
2404         DISP ERRM$;
2406      END SELECT
2408      DISP " ....CONTINUE to try again."
2410      PAUSE
2412      Filename$=""
2414      Diskdrive$=""
2416      GOTO Selectdrive
2418      !
2420      RETURN
2422 !
2424   SUBEND
2426 !
2428 !*************************************************************************
2430 !
2432   SUB Enterfilename(Ac$)
2434 Enterfilename:  ! Original: 13 Nov 1984
2436              ! Revision: 10 Dec 1990  includes HFS directories
2438      OPTION BASE 1
2440      COM /Files/ Diskdrive$[20],Filename$[14],Ms_path$[500]
2442      COM /Interrupts/ INTEGER Intr_prty
2444      INTEGER I,Ascii_num,Maskflag,Namelength
2446      DIM Test$[256],Hfs_temp$[161]
2448      Namelength=10
2450      IF LEN(Ms_path$)>0 THEN OUTPUT KBD USING "K,#";"#"&Ms_path$&"H"
2452      DISP " ENTER HFS directory PATH (no file)";
```

```
2454      IF Ac$<>"PATH" THEN
2456         DISP ", ENTER / for HFS ROOT or null for LIF...";
2458      END IF
2460      LINPUT Hfs_temp$
2462      Hfs_temp$=TRIM$(Hfs_temp$)
2464      IF LEN(Hfs_temp$)>0 THEN
2466         IF LEN(Hfs_temp$)>1 AND Hfs_temp$[LEN(Hfs_temp$);1]<>"/" THEN
2468            Hfs_temp$=Hfs_temp$&"/"
2470         END IF
2472         IF LEN(Hfs_temp$)=1 THEN Hfs_temp$=""
2474         Namelength=14
2476      END IF
2478      IF Ac$="PATH" THEN
2480         Ms_path$=Hfs_temp$
2482         SUBEXIT
2484      END IF
2486      IF LEN(Filename$)>0 THEN OUTPUT KBD USING "K,#";"#"&Filename$&"H"
2488 Efn: !
2490      DISP " ENTER the FILE NAME ... ";
2492      SELECT Ac$
2494      CASE "CAT"
2496         DISP "(ENTER CAT mask* or ENTER null to CAT)";
2498      CASE "ABORT"
2500         DISP "(ENTER null to ABORT) ";
2502      CASE "VALID"
2504         DISP "(must be a VALID name!) ";
2506      END SELECT
2508      LINPUT Test$
2510      Test$=TRIM$(Test$)
2512      IF LEN(Test$)=0 AND Ac$="VALID" THEN GOTO Enterfilename
2514      IF LEN(Test$)=0 THEN Abortline
2516      IF LEN(Test$)>Namelength THEN
2518         BEEP
2520         DISP "ERROR in NAME ENTRY - max ";Namelength;" chars, you have ";
2522         DISP LEN(Test$);" "
2524         WAIT 1.8
2526         OUTPUT 2 USING "K,#";"#"&Test$&"H"
2528         GOTO Efn
2530      END IF
2532      IF POS(Test$,"*")>1 THEN
2534         Test$=Test$[1,POS(Test$,"*")-1]
2536         Maskflag=1
2538      ELSE
2540         Maskflag=0
2542      END IF
2544      FOR I=1 TO LEN(Test$)
2546         Ascii_num=NUM(Test$[I])
2548         SELECT Ascii_num
2550         CASE 65 TO 90,95,97 TO 122,48 TO 57
2552            !Allowed characters
2554         CASE ELSE
2556            BEEP
2558            DISP "ERROR in NAME ENTRY--ILLEGAL CHARACTERS, TRY AGAIN."
2560            WAIT 1.8
2562            OUTPUT 2 USING "K,#";"#"&Test$&"H"
2564            GOTO Efn
```

```
2566          END SELECT
2568        NEXT I
2570        IF Maskflag THEN
2572          Filename$=Test$&"*"
2574        ELSE
2576          Filename$=Test$
2578        END IF
2580        Ms_path$=Hfs_temp$
2582        SUBEXIT
2584 Abortline:Filename$=""
2586        IF Ac$="CAT" THEN Ms_path$=Hfs_temp$
2588        SUBEXIT
2590    SUBEND
2592    !
2594    !*****************************************************************
2596    !
```

BL-114A
(5-90)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
NIST 3978

2. PERFORMING ORGANIZATION REPORT NUMBER
B91-0277

3. PUBLICATION DATE
DECEMBER 1991

4. TITLE AND SUBTITLE

AWAMS Users Manual

5. AUTHOR(S)

S.M. Chesnut, N.G. Paulter

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
BOULDER, COLORADO 80303-3328

7. CONTRACT/GRANT NUMBER

8. TYPE OF REPORT AND PERIOD COVERED

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

Army TMDE Support Group, Bldg. 5435
Redstone Arsenal, Alabama 35898

10. SUPPLEMENTARY NOTES

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

The theory and operation of an upgraded version of the NIST Automatic Waveform Analysis and Measurement System is described. This system, the AWAMS, was commissioned by the Army Primary Standards Laboratory to facilitate measurement comparability with NIST. The AWAMS has been installed at the Redstone Arsenal, Alabama.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)

acquisition; calibration; deconvolution; jitter; pulse parameters; waveform

13. AVAILABILITY

X	UNLIMITED
	FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).
	ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.
X	ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
323

15. PRICE
A14

ELECTRONIC FORM


AMERICAN BINDERY
2
5-94