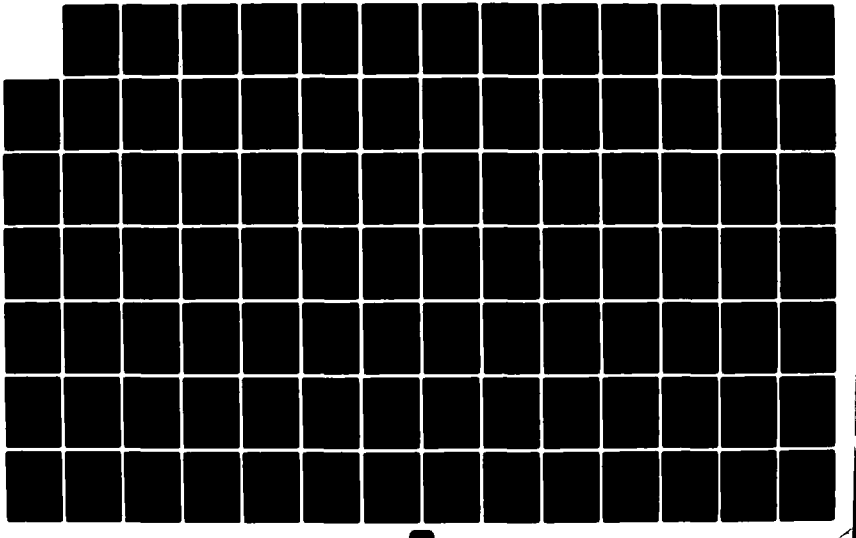
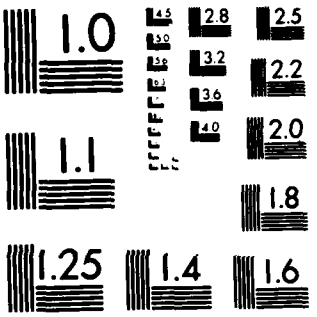
AD-A126 980 A MICROCOMPUTER-BASED SAMPLING DIGITAL VOLTMETER(U) 1/2
NAVAL RESEARCH LAB WASHINGTON DC R E SCOTT ET AL.
31 MAR 83 NRL-MR-4872

UNCLASSIFIED F/G 14/2. NI





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

NRL Memorandum Report 4872

A Microcomputer-Based Sampling Digital Voltmeter

RICHARD E. SCOTT, JR. AND JAMES D. GEORGE

Acoustical Systems Branch
Underwater Sound Reference Detachment
Naval Research Laboratory
P.O. Box 8337
Orlando, Florida 32856

March 31, 1983

NAVAL RESEARCH LABORATORY
Washington, D.C.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NRL Memorandum Report 4872	2. GOVT ACCESSION NO. AD-A126 800	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Microcomputer-Based Sampling Digital Voltmeter		5. TYPE OF REPORT & PERIOD COVERED Interim report on a continuing problem
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Richard E. Scott, Jr. and James D. George		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Underwater Sound Reference Detachment Naval Research Laboratory P.O. Box 8337, Orlando, FL 32856		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Element: 62711N Project NRL Work Unit: (59)-0593
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Sea Systems Command (NAVSEA 63R12) Washington, DC 20362		12. REPORT DATE 31 March 1983
		13. NUMBER OF PAGES 109
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This work was performed in the Acoustic Metrology Program sponsored by the Naval Sea Systems Command (SEA63R12).

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Vector voltmeter	Distortion measurement	Kaiser window
True rms	A/D converter	Microcomputer
Phase measurement	SFDFT	PDP-11
		IEEE-488 bus

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes a sampling digital voltmeter developed at the Underwater Sound Reference Detachment of the Naval Research Laboratory. It consists of a PDP-11/23 microcomputer with up to three channels of analog-to-digital converters capable of making simultaneous measurements on pulsed or continuous sinusoids at frequencies up to 100 kHz. The voltmeter is controlled by ASCII commands on the IEEE-488 bus. Statistical parameters (mean, peak, standard deviation, and second joint moment) are computed using numerical integration. Sinusoidal parameters (amplitude, phase, and harmonic

(continued)

(Item 20 continued)

distortion) are computed using single-frequency discrete Fourier transforms (SFDFT). The digital filtering resulting from the use of the SFDFT can be further enhanced by optional data sequence weighting (multiplying the sequence by a given windowing sequency). Computation times may be reduced by optional subsequence averaging. Complete assembly language listings for the voltmeter program are included and described.

CONTENTS

A MICROCOMPUTER-BASED SAMPLING DIGITAL VOLTMETER

INTRODUCTION

Acoustical measurement systems at the Underwater Sound Reference Detachment (USRD) of the Naval Research Laboratory need a specialized multiple-channel voltmeter with the capability to measure voltage, relative phase between channels, and harmonic distortion for pulsed sinusoids at frequencies between 0.1 Hz and 100.0 kHz. To answer this need, a sampling digital voltmeter was developed around an LSI-11/23 microcomputer, using three analog-to-digital (A/D) converters and an IEEE-488 interface to serve as the communications link with any appropriate host computer. Using this voltmeter, a "simple" computer-controlled measurement system can then be configured with the following auxiliary equipment as shown in Fig. 1: synthesizers to generate a sinusoidal test signal and a coherent sampling signal, and programmable amplifiers or attenuators as needed. (The voltmeter has no intrinsic autoranging capability.)

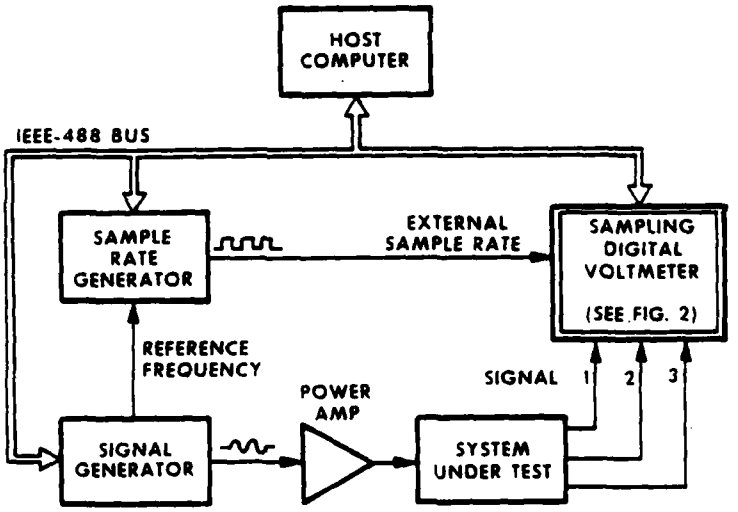


Fig. 1 - Typical measurement system using the microcomputer-based sampling voltmeter.

APPLICATIONS

The two immediate applications for such a voltmeter were for the USRD's Anechoic Tank Facility (ATF) and Low-Frequency Facility (LFF) measurement systems.

The ATF high-power measurement system requires a three-channel voltmeter to measure simultaneous projector drive voltage, drive current, and receiver output (hydrophone voltage). Measurements of relative phase between voltage and current and the harmonic distortion in all three channels are necessary. The practical frequency range is 2 to 100 kHz; the mode of operation is usually pulsed.

The LFF system requires a two-channel voltmeter to measure the simultaneous voltage outputs of two hydrophones, generally a probe standard and an unknown hydrophone. Relative acoustic phase is required, and harmonic distortion is a useful parameter to monitor. The practical frequency range is 0.1 Hz to 4 kHz; the mode of operation is continuous wave (cw).

The voltmeter was, therefore, conceived to be a device independent of (and transportable between) specific measurement systems at the USRD, with the possible exception being that the A/D converters might vary with application (e.g., a need to operate at higher sampling rates) and are otherwise functionally identical plug-in modules. To get the widest possible range of potential host computers, communication through the IEEE-488 bus was chosen. Another requirement was that all external hardware, including the sample-rate generator, be controlled by the host computer for maximum flexibility. Therefore, all external functions, including automatic ranging of the voltmeter, are controlled by the host.

Since initiating and implementing this signal processing sampling voltmeter, two other similar instruments have been described or released: the low-frequency sampling voltmeter by the National Bureau of Standards [1] and the sampling network analyzer by the Dranetz Company [2]. The three instruments share the sampling ideas but differ in the features and algorithms implemented.

THEORY OF OPERATION

The voltmeter consists of a microcomputer containing up to three A/D converters with direct memory access (DMA) capability (Fig. 2). This configuration allows the computer to digitize up to three input waveforms simultaneously, sampling them when triggered by a common external sample-rate generator. Where simultaneous measurement of all channels is not desired, separate or delayed sampling signals can be given to individual A/D converters. The present voltmeter is designed to digitize three waveforms of up to 1024 samples each; these buffers could be expanded to 3900 samples (see Appendix F).

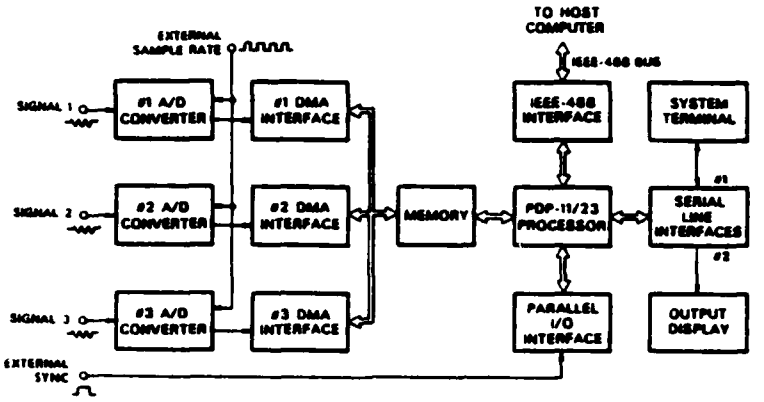


Fig. 2 - Block diagram of the microprocessor-based sampling voltmeter.

The DMA interface loads 12-bit digitized values directly into memory at rates up to 100 kHz per channel. The individual sampled waveforms can be transmitted to the host for processing; alternately, the slave can compute sinusoidal and statistical parameters from these waveforms and return them to the host. Parameters are computed for the fundamental frequency. (Throughout this report "fundamental" frequency refers to the sinusoidal test frequency being be measured, no matter how many cycles of this frequency are included in a measured sequence.)

The sinusoidal quantities amplitude, phase, and harmonic distortion are computed for each channel using single frequency discrete Fourier transforms (SFDFT). The statistical quantities average, standard deviation, and the positive and negative extrema are obtained for each channel to characterize noisy or nonsinusoidal signals. In voltmeter terms, these statistical parameters are the dc, true ac rms, and positive and negative peak voltages. To measure the coherence or correlation between channels, the second order joint moment statistic, E(xy), can be computed. For the case of the two signals, x and y; proportional to current and voltage, the joint moment is proportional to power. The covariance and cross correlation can be readily obtained from these parameters.

COMPUTATION PROCEDURES

Sampling Criteria

In the usual case of interest (pulsed sinusoids), sampling errors are minimized through coherent sampling, which requires an integer number of both test frequency cycles and sampling periods:

$$\frac{m}{f_T} = \frac{n}{f_S} \quad , \tag{1}$$

where f_T is test frequency
f_S is sampling frequency
n is samples per sequence (integer)
m is cycles per sequence (integer).

If the sampling frequency,

$$f_S = \frac{n}{m} f_T \quad , \tag{2}$$

is greater than twice the test frequency, the Nyquist theorem is satisfied, and satisfactory results will be obtained. If the necessary sampling rate is too fast for the A/D converter to operate properly, undersampling can be used; by choosing fewer than two samples per cycle while retaining an integer number of samples and cycles per sequence, an effective higher sampling rate is attained. A comparison of the oversampled waveform shown in Fig. 3 with the undersampled waveform in Fig. 4 clearly demonstrates how undersampling can synthesize a waveform (Fig. 5) effectively sampled at a higher rate, although with fewer cycles overall. Undersampling will only work properly on periodic waveforms, however. Computed data in Table 1 demonstrates that peak voltage and harmonic distortion may be affected slightly by undersampling.

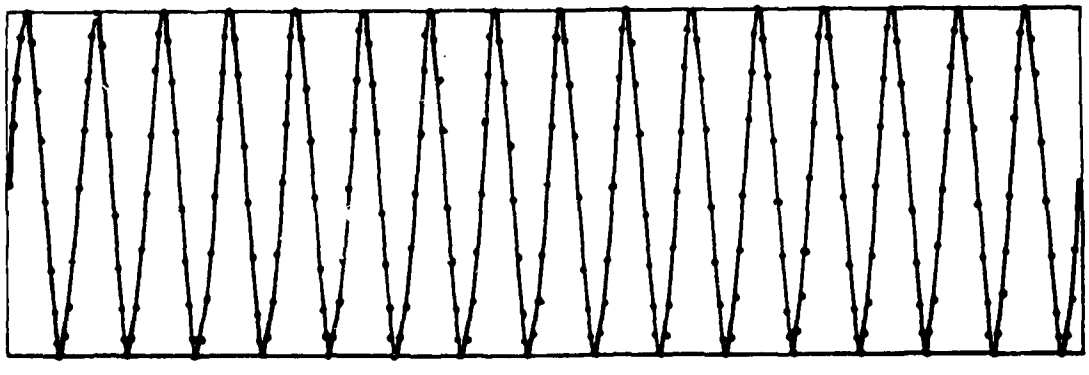


Fig. 3 - Steady-state sinusoidal waveform sampled for 16 cycles at 17 samples per cycle: 272 samples (oversampling).

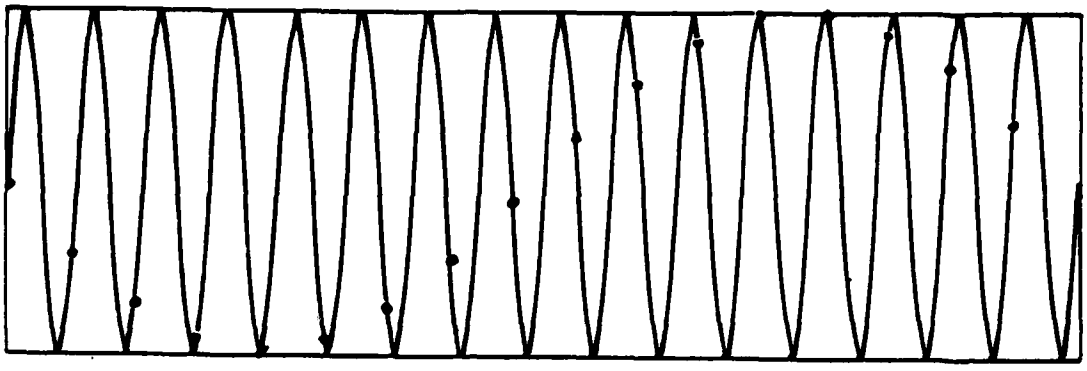


Fig. 4 - Steady-state sinusoidal waveform sampled for 16 cycles at 1.0625 samples per cycle: 17 samples (undersampling).

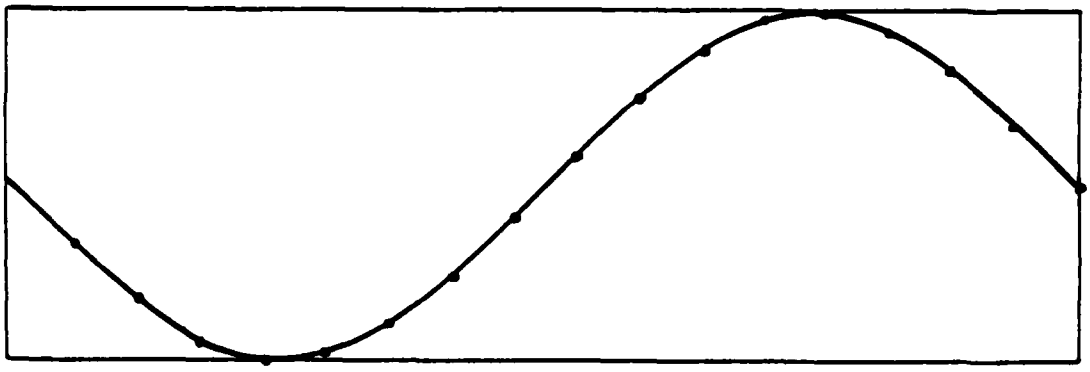


Fig. 5 - Steady-state sinusoidal waveform synthesized from the 17 samples shown in Fig. 5.

Table 1 - Comparison of oversampled and undersampled measurements.

COMPUTED PARAMETER	OVER SAMPLING	UNDER SAMPLING
RMS VOLTAGE (STD DEV)	1.047	1.047
PEAK VOLTAGE	1.485	1.492
DC VOLTAGE (MEAN)	-0.013	-0.013
TOTAL POWER	1.102	1.102
RMS VOLTAGE (DFT)	1.047	1.047
RELATIVE PHASE	0	0
HARMONIC DISTORTION	0.100	0.095

Normal: 2 cycles at 26 samples per cycle (52 samples total).
Undersampled: 25 cycles at 1.04 samples per cycle (52 samples total).

Sinusoidal Parameters

For sinusoidal testing, the signals are of the form

$$y(t) = \sqrt{2}A_m \cos(2\pi f_T t + \phi_m) + \sum_{k=1}^{\ell} \sqrt{2}A_{km} \cos(2\pi k f_T t + \phi_{km}) + n(t) \quad , \tag{3}$$

where the terms are, respectively, the fundamental, the harmonics, and the additive noise.

Least squares estimates of the rms amplitude, A_m, and phase, ϕ_m, of the fundamental are obtained from

$$\hat{A}_m = \sqrt{2}\left[Re^2(Y_m) + Im^2(Y_m)\right]^{1/2} \quad , \tag{4}$$

and

$$\hat{\phi}_m = \tan^{-1}\left[Im(Y_m)/Re(Y_m)\right] \quad , \tag{5}$$

where

$$Y_m = \frac{1}{n} \sum_{i=0}^{n-1} y_i \exp(-j2\pi im/n) \quad , \tag{6}$$

which is the mth component of the DFT of the n-point sequence y_i containing m cycles of the fundamental; that is, the single frequency discrete Fourier transform (SFDFT). With coherent sampling and in the absence of noise, $\hat{A}_m$ and $\hat{\phi}_m$ are exact.

Harmonic distortion, D, is defined in terms of the ratio of power in the harmonics to power in the fundamental.

$$D = \left[\sum_{k=2}^{\ell} A_{km}^2/A_m^2\right]^{1/2} . \tag{7}$$

Since the sequences are relatively short and few lines are required, Goertzel's algorithm [3,4] is an efficient implementation of the SFDFT.

Digital Filtering

The evaluation of the DFT at a single frequency as in Eq. (6) can be interpreted as a digital filter [5] with a corresponding enhanced signal-to-noise ratio. However, the filter quality, the skirts and side lobes, can be significantly improved by weighting or windowing the data sequence and then doing the SFDFT. A particularly effective choice of data weighting sequence is the Kaiser I_o-sinh window [6]. An example of a weighted data sequence is shown in Fig. 6, and the corresponding digital filter responses are shown in Fig. 7.

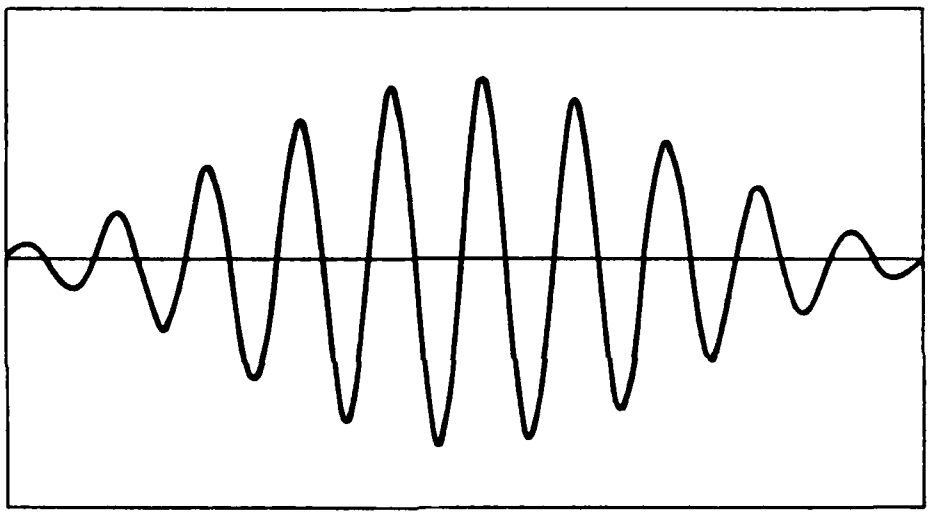


Fig. 6 - Sinusoidal waveform of 10 cycles (product with Kaiser window with 40 dB of spectral side lobe attenuation).

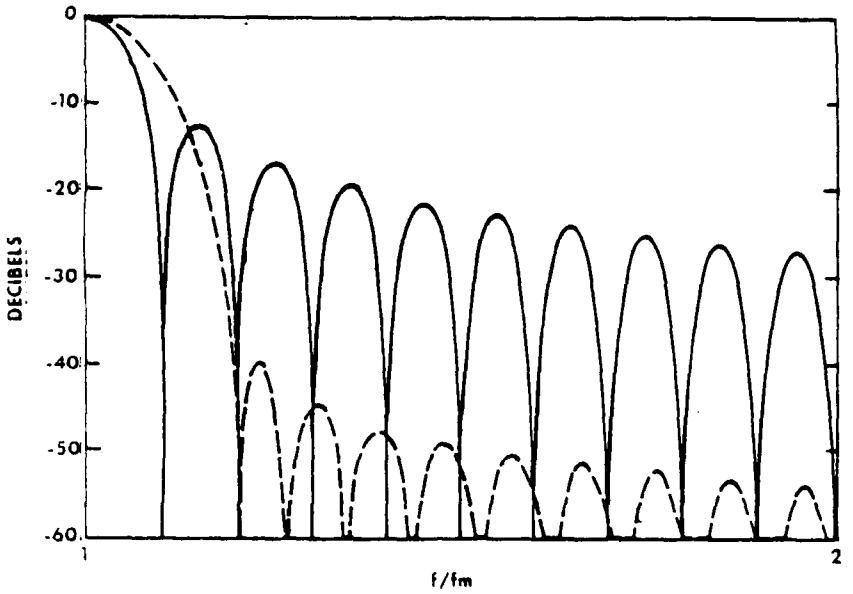


Fig. 7 - Comparison of the filter effect of an unweighted SFDFT (solid line) with that of a SFDFT with the Kaiser window with 40 dB of side lobe attenuation (dashed line). The frequency axis is normalized to the center frequency f_m which corresponds to the mth spectral line. The data sequence includes 10 cycles.

The filters are centered on the mth spectral line or test frequency, f_T. Following convention, the filter bandwidth is defined as the main lobe width or the distance between the minima adjacent to the mth line. For the unweighted SFDFT, the fractional bandwidth or main lobe width is

$$\left(\frac{\Delta f}{f_m}\right)_{\text{Rectangular}} = \frac{2}{m} \quad , \tag{8}$$

where m is the number of cycles of f_T in the data sequence.

The reduced side lobes of the Kaiser filter are achieved at the expense of an increased main lobe width

$$\left(\frac{\Delta f}{f_m}\right)_{\text{Kaiser}} = \left(\frac{2}{m}\right) \frac{6(R + 12)}{155} \quad , \tag{9}$$

where R is the attenuation in dB of the first side lobe. Figure 8 illustrates the increase in filter main lobe width as the side lobes are reduced.

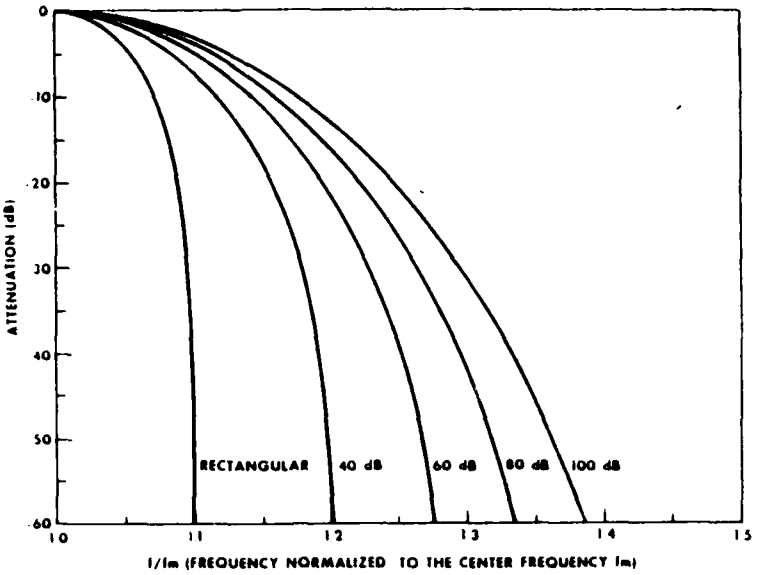


Fig. 8 - Comparison of filter main lobes for unweighted SFDFT and Kaiser-weighted SFDFT with 40 to 100 dB side lobes. The side lobes are not plotted for clarity. The data sequence includes 10 cycles (m = 10).

The data window or weighting sequence, w_i, is computed in the host and transferred to the voltmeter where the window is applied in estimating amplitude and phase using the SFDFT of the weighted sequence

$$Y_m = \frac{1}{n} \sum_{i=0}^{n-1} w_i y_i \exp(-j2\pi im/n) \quad . \tag{10}$$

The amplitude estimate, A_m, is corrected by the window scale factor

$$\bar{w} = \frac{1}{n} \sum_{i=0}^{n-1} w_i \quad . \tag{11}$$

Table 2 illustrates that the Kaiser window does not alter or bias the amplitude and phase estimates at the test frequency. However, the harmonic distortion, particularly for low distortion, is biased by the window; larger sidelobe attenuations reduce this bias by reducing the amplitude of the harmonics created by the window.

Table 2 - Amplitude, phase and harmonic distortion estimates with and without a Kaiser window.

WINDOW	MEASURED DATA			SIMULATED DATA		
	$\hat{A}_m$ RMS AMPLITUDE	$\hat{\phi}_m$ PHASE DEGREES	100D PERCENT HARMONIC DISTORTION	100D PERCENT HARMONIC DISTORTION		
				0%	0.1%	1%
No Window	1.054	-123.3	0.060	0.000	0.100	1.000
- 30 dB Side Lobe	1.054	-123.3	0.071	0.032	0.073	0.972
- 40 dB Side Lobe	1.054	-123.3	0.077	0.018	0.084	0.984
- 50 dB Side Lobe	1.054	-123.3	0.082	0.009	0.092	0.992
- 60 dB Side Lobe	1.054	-123.3	0.087	0.004	0.097	0.997
- 70 dB Side Lobe	1.054	-123.3	0.097	0.002	0.099	0.999
- 80 dB Side Lobe	1.054	-123.3	0.093	0.001	0.099	0.999
- 90 dB Side Lobe	1.054	-123.3	0.095	0.000	0.100	1.000
-100 dB Side Lobe	1.054	-123.3	0.097	0.000	0.100	1.000

Computations performed on 20 cycles of data at 32 points per cycle. Measured data was the output of a synthesizer; simulated data was a computed sinewave with distortion added to the second harmonic as labeled.

When long sequences or long integration times are used to combat noise, the computation time can be significantly reduced by averaging. This efficiency is possible if the total sequence consists of two or more subsequences with an integer number of samples and cycles per subsequence. The simplest example is a data sequence of m cycles of the fundamental test sinusoid with each cycle containing n samples--this data sequence can be compressed from m cycles to one cycle by averaging. Both the SFDFT and windowed SFDFT computations can be speeded up through averaging. (The window is applied before averaging.)

Statistical Parameters

For signals which may be arbitrary periodic waveforms or noise-like data, statistical parameters are more appropriate in characterizing the data than are the SFDFT derived amplitude and phase (although naturally these parameters can be computed for sinusoidal signals as well).

The voltage extremes or range of the waveform are obtained by inspection of the data sequence, and both the maximum and minimum voltages are found.

The average or dc voltage is computed with

$$y_{dc} = \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \quad . \tag{12}$$

The standard deviation or true ac rms is obtained from

$$y_{rms} = \sigma_y = \sqrt{\sum_{i=1}^{n} \frac{1}{n} (y_i - \bar{y})^2} \quad . \tag{13}$$

The choice of factor 1/n instead of 1/(n-1) is to satisfy the Parseval relationship so that the rms amplitude of an ideal sine wave is the same whether estimated with the SFDFT or with the true rms methods.

The second joint moment

$$E(xy) = \frac{1}{n} \sum_{i=1}^{n} x_i y_i \quad , \tag{14}$$

is analogous to power if the data sequences x_i and y_i are proportional to current and voltage. Power measurement was the motivation for including this joint moment estimate. The total ac power, P_{ac}, is computed using

$$P_{ac} = E(xy) - \bar{x}\,\bar{y} \quad . \tag{15}$$

All of the summations in Eqs. (10) through (15) can be interpreted as Euler-Maclaurin approximations to integrals of the continuous time functions. For functions that can be represented as a finite number of harmonically-related sinusoids, this simple approximation is exact [7].

SOFTWARE FUNCTIONS

The voltmeter software is organized around a monitor routine awaiting input from the host via the IEEE-488 interface (as shown in Fig. 8). The input consists of ASCII strings of variable length--the first two characters being a code for setup or control. Strings are terminated by an end-or-identify (EOI) sent with the last character.

"Setup" codes are accompanied by an encoded byte string specifying the value for a certain parameter, such as the programmable gain for a specific A/D converter. Setup strings tell the program what values are to be used, but do not perform any operation such as physically setting the gain of an A/D converter. No default parameters are present within the program; all parameters must be specifically loaded.

"Control" codes (two-byte strings) command the voltmeter to perform certain functions; i.e., make an A/D conversion, compute certain parameters, transfer data to the host, etc.

The voltmeter can be requested to compute and return various parameters as requested, or to return the original sampled waveform to the host for special processing not supported by the voltmeter (plotting, for example). It is necessary to point out that although commands are transferred to the voltmeter as ASCII byte strings, data is received from the voltmeter in PDP-11 format (two bytes forming the 16-bit integers for sampled data, or four bytes floating point variables for all computed parameters). This was done in the interest of faster data transfer and maintaining original data precision for NRL-USRD applications. This feature prevents full compatibility with non-PDP 11 host processors, although subroutines to encode integer and floating-point variables exist as part of the display functions, and the program could be modified accordingly.

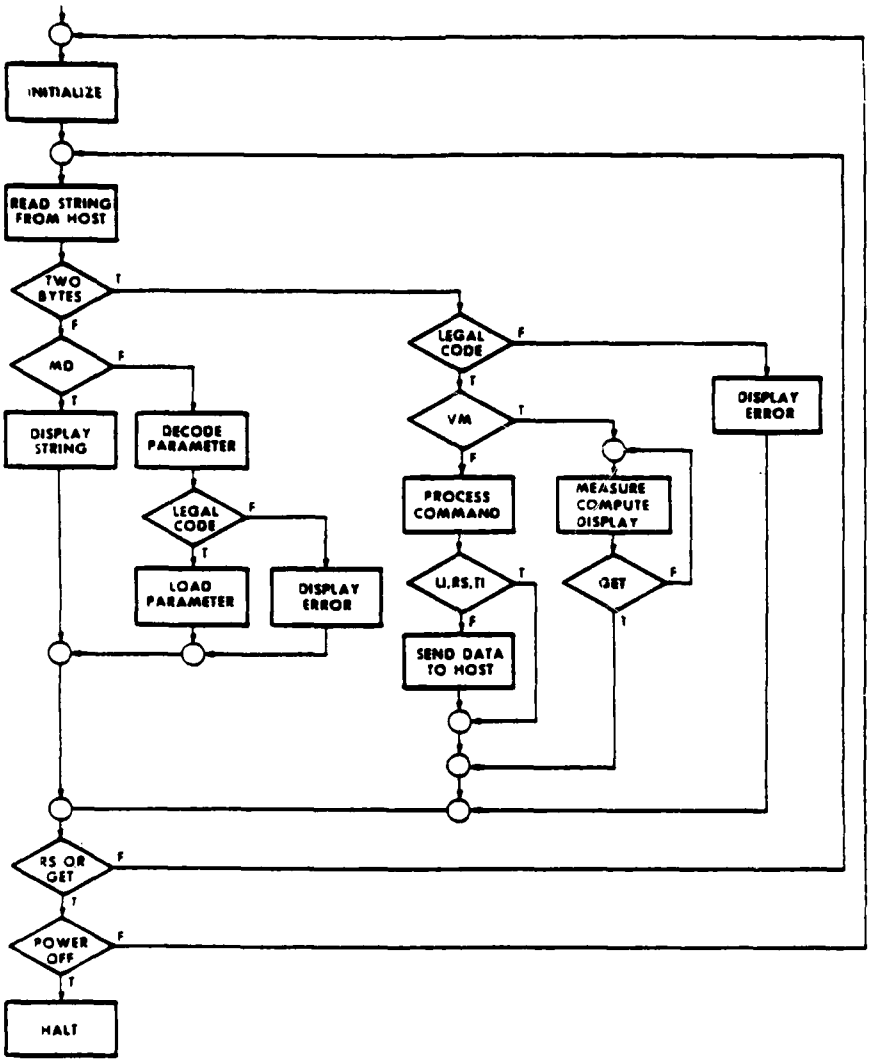


Fig. 9 - Flow chart of the basic monitor logic in the voltmeter software.

A special control code (VM) allows the voltmeter to measure, compute, and display as its own pace without host control; the host regains control by generating a group-execute-trigger (GET) on the IEEE-488 bus. The voltmeter displays data on either the console terminal or a 24-character plasma display. Display styles allowed are: count of A/D conversions and errors; voltage, phase, and distortion of one specific channel; peak voltage of all three channels; or voltages of two channels and the phase between the two.

The computation commands follow this sequence: the buffer of 12-bit digitized integer values is converted to floating-point numbers representing voltages within the A/D converter range (±10 V); a window function is applied, if desired; the data sequence in the buffer is averaged into a smaller subsequence, if desired; the specific computation is made; the results are scaled to account for the A/D converter gain and corrected for amplitude

change due to the window function, if used; and the data are transmitted to the host. The use of windowing and subsequence averaging seems not very useful for the computation of nonsinusoidal parameters.

The four primary commands are: "D" (compute voltage, phase, and distortion from SFDFT); its subset, "C" (the same without distortion), used for more rapid computation where distortion is not needed; "E" (total rms, peak, and dc voltage); and a similar subset, "F" (absolute peak only), a rapid tool for system autoranging or overload detection. Relative computation times are shown in Table 3. Note that these times refer to a PDP-11/23 computer with both the KEF-11 floating-point processor and the new FPF-11 floating-point hardware. The FPF-11 offers as much as a factor of 5 improvement in computation speed.

Table 3 - Sample computations measured for a PDP-11/23 with the KEF-11 and FPF-11 floating-point processors.

	COMPUTATION TIME IN SECONDS							
	NO SUBSEQUENCE AVERAGING				WITH SUBSEQUENCE AVERAGING			
	NO WINDOW		WITH WINDOW		NO WINDOW		WITH WINDOW	
COMPUTATION STYLE	KEF-11	FPF-11	KEF-11	FPF-11	KEF-11	FPF-11	KEF-11	FPF-11
SINUSOIDAL PARAMETERS								
Voltage, Phase	0.494	0.149	0.570	0.172	0.210	0.098	0.293	0.132
Voltage, Phase, Distortion	2.627	0.506	2.632	0.531	0.383	0.133	0.458	0.165
NONSINUSOIDAL PARAMETERS								
rms, Peak, dc Voltage	0.350	0.151	-	-	0.198	0.100	-	-
Peak Voltage	0.189	0.096	-	-	-	-	-	-
Total Power	1.488	0.477	-	-	0.927	0.373	-	-

Average of 100 measurements of 20 cycles of data at 40 points per cycle (800 points). Distortion computed from first 7 harmonics. Subsequence averaging to 1 cycle. Times include data transfer on IEEE-488 bus (about 0.016 second).

Collateral computation commands include the ability to compute the total rms voltage and phase of a specific harmonic and to compute the power parameters second joint moment (ac and dc power) and covariance (ac power). The power parameters are the total power generated by all harmonics, and it is assumed that the host will scale the data properly to account for the calibration of the current measuring circuitry.

Further details of the software are to be found in the appendices and the liberally-commented source code located there. Appendix C lists various error and status messages; Appendix D gives listings for sample host routines to drive the voltmeter; and Appendix E includes complete listings of all software modules loaded into the voltmeter, with external discussion where appropriate.

SUMMARY

A multiple-channel microprocessor-based sampling voltmeter has been developed for use in the USRD measurement systems. This voltmeter is capable of measuring amplitude, phase, and distortion, and of using several computational algorithms to derive these results as the application requires. The original voltmeter has been used successfully in the Anechoic

Tank Facility as a high-voltage impedance measurement system, and this more powerful version has been greatly expanded to be used in Low-Frequency Facility, or other measurement systems as needed.

ACKNOWLEDGMENTS

The authors would like to acknowledge R.W. Anderson and R.W. Luckey, both of the USRD, and National Instruments, Inc., for their assistance in making this device function on the IEEE-488 interface bus, and C.K. Brown (USRD) for his assistance in incorporating the voltmeter into an operating electronic measurement system. This work was performed in the Acoustic Metrology Program sponsored by the Naval Sea Systems Command (SEA63R12).

REFERENCES

1. Barry A. Bell, Bruce F. Field, Thomas Kibalo, "A Fast Response Low-Frequency Sampling Voltmeter," National Bureau of Standards Tech Note 1159, National Bureau of Standards, Gaithersburg, MD, Aug 1982.

2. Dranetz Engineering Laboratories, Dranetz Model 3100 Sampling Network Analyzer Bulletin 3100, Edison, NJ.

3. G. Goertzel, "An Algorithm for the Evaluation of Finite Trig Series," *American Mathematical Monthly,* Vol. 65, 34-35 (1958).

4. G. Goertzel, *Mathematical Methods for Digital Computers, Vol I,* A. Ralston and H.S. Wolf, eds., John Wiley & Sons, New York, NY, 1960 (10th printing 1967), Ch. 24, 258-262.

5. F.J. Harris, "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform," *Proceedings of the IEEE,* Vol. 66, No. 1, 51-83 (1978).

6. J.F. Kaiser and R.W. Schafer, "On the Use of the I_o-Sinh Window for Spectrum Analysis," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* Vol ASSP-28, No. 1, 105-107 (1980).

7. Philip J. Davis and Philip Rabinowitz, *Numerical Integration,* Blaisdell Publishing Company, Waltham, Mass., 1967, Sec. 2.9.

8. National Instruments, "GPIB11V-1 Operating and Service Manual, April 1980, Part No. 320002-01; and "Software Reference Manual for National Instruments GPIB11-Series Interface Cards," September 1981, Part No. 310001-01.

APPENDIX A

Voltmeter Hardware

The hardware chosen as the heart of the voltmeter was the Digital Equipment Corporation (DEC) PDP-11/2 microcomputer (later upgraded to the PDP-11/23 with a KEF-11 floating-point processor), because it was a device that the USRD had both the experience and facilties to handle.

For A/D converters, ADAC Corp. models with DMA interfaces were chosen: model 1012, whose maximum sampling rate of 35 kHz was satisfactory for the LFF voltmeter, and model 1012AD, whose maximum sampling rate of 100 kHz was necessary for the ATF voltmeter.

Other interface modules used were:

MXV11-A	(DEC) Multifunction module: 2-serial line units 16K words RAM memory 4K words ROM memory (Intel 2732 EPROMs)
GPIBV11-1	(National Instruments) IEEE-488 bus interface
DRV11	(DEC) 16-bit parallel I/O card

The parallel I/O card is used only in pulsed systems to synchronize the measurements with the beginning of a toneburst.

(BLANK PAGE)

APPENDIX B

Voltmeter Commands

SETUP COMMANDS (# is channel number): These consist of a 2-byte code and a decimal integer string for the parameter value.

AE	Number of times to automatically retry a measurement if an A/D error occurs
AI	Which A/D-DMA device generates an interrupt when the data transfer is complete (1, 2, 3)
AN	Which A/D converters are to be used; additive, where 1 = #1, 2 = #2, 4 = #3, (7 = all three)
AV	Averaging mode (0 = no averaging, >0 = number of cycles in the averaged subsequence)
CT	Computation type (for future use)
DD	Display device: serial line 1 (console) or 2 (plasma display)
DS	Display style: 0 (conversion count), 1-3 (voltage, phase, distortion of channels 1-3), 4 (peak voltage in all 3 channels), 5 (voltages in channels 1-2 and phase between)
FC	Sample (clock) frequency in Hz
FS	Signal frequency in Hz
GV	Gated system (0 = not gated, 1 = use parallel I/O card to synchronize with toneburst)
G#	Programmable gain for A/D converter (1, 2, 5, 10)
HA	Number of harmonics used to compute distortion
HC	Which harmonic to compute (see B# - CONTROL COMMANDS)
MD	Transfer message to display, a byte string of up to 40 characters
NC	Number of cycles per sequence
PA	Number of points to convert per sequence
PR	Number of points per sequence to return to host

PS First point to return to host. In practice, the first few converted points are bad, therefore, PA = PR + PS points are converted and PR points are computed

P# Port (channel) for A/D Multiplexor (0-15)

WM Window mode (0 = don't use window, 1 = use window)

CONTROL COMMANDS (2-byte control codes; # is channel number)

B# Compute and return voltage and phase of the harmonic defined by HC parameter

CW Compute a sine wave for test purposes

C# Compute and return voltage and phase of the fundamental

D# Compute and return voltage, phase, and distortion of the fundamental

E# Compute and return statistical data (rms, dc, positive and negative voltage extremes)

F# Compute and return absolute peak voltage

LI List setup parameters on the display

ME Perform A/D conversion and return status

PO Compute and return total power (second joint moment)

RP Transfer setup parameters to host

RS Clear conversion and error counters

R# Read back a data sequence

RW Read back the average of a window and the weighted sequence

SJ Compute and return total power (second joint moment) and covariance (ac power)

TI Measure the time used in data transfer over the IEEE-488 bus

VM Set to free-running display voltmeter mode

WI Prepare voltmeter to read a computed window sequence from the host

SPECIAL IEEE-488 COMMANDS

EOI End or identity - must coincide with last character sent by the host

GET Group execute trigger - command will interrupt whatever the voltmeter is doing and reinitialize

APPENDIX C

Voltmeter Status Messages

Status is returned as an integer variable that is the voltmeter's acknowledgment of a measure command. It refers to either the legality of the command or the success of executing it.

STATUS	MEANING
1	Normal return
-1	A/D error in channel 1
-2	A/D error in channel 2
-4	A/D error in channel 3
-10	DMA error in channel 1
-20	DMA error in channel 2
-40	DMA error in channel 3
-100	A/D setup error
-101	Too many points for program buffer

Errors -1 through -40 are added together in case of multiple errors, and thus cover all numbers between -1 and -77.

The voltmeter processor will halt for the following severe errors:

ADDRESS	REASON FOR HALT
002036	Floating-point exception (can proceed)
002054	Function (sine, atan, sqrt) error (can proceed)
003370	Too many samples to compute (can proceed)
006224	Illegal display mode (can proceed)
013236	IEEE-488 interface illegally strapped (fatal)

(BLANK PAGE)

APPENDIX D

Voltmeter Control Subroutines

The following describes the protocol necessary and the available general-purpose subroutines for controlling this sampling voltmeter from a host processor. The National Instruments IEEE-488 bus driver and its software interface IBUP are a prerequisite [8].

Listings for the following subroutines (written at NRL-USRD) are included in this appendix.

1. Primitive subroutines

RDLSI	Poll the bus and if successful read a byte string from the voltmeter
WRLSI	Write a byte string to the voltmeter

2. Higher-level I/O subroutines

WRLPA	Send an integer parameter to voltmeter
WRLVM	Send an integer array to voltmeter
WRMEA	Send measurement command to voltmeter, wait an appropriate moment, enable specific clock synthesizer, receive status from voltmeter, disable specific clock synthesizer

3. Example of voltmeter I/O

METST	Test routine to command a measurement, read back a data sequence, and convert to volts

```
FORTRAN IV-PLUS V3.0              21:07:42     25-Mar-82              Page 1
RDLSI.FTN;101              /TR:BLOCKS/WR

0001                SUBROUTINE RDLSI (IADR,IBUF,ILEN,ISW)
          C
          C         SUBROUTINE TO READ A BYTE STRING FROM THE LSI VOLTMETER.
          C
          C         ARGUMENTS:
          C                 IADR = GPIB BUS ADDRESS
          C                 IBUF = BYTE-BUFFER TO LOAD WITH STRING
          C                 ILEN = LENGTH OF STRING EXPECTED
          C                 ISW  = STATUS OF READ PROCESS
          C                          ISW = 1 (SUCCESSFUL)
          C
          C         AUTHOR: R. E. SCOTT, JR. (NRL/USRD); VERSION: JANUARY 1982
          C
0002                BYTE IBUF(ILEN)
0003                INTEGER GPIB
          C
0004      100       CONTINUE
          C
          C         TEST FOR SRQ (CURRENTLY NOT WORKING WITH LSI)
          C         J=GPIB(13)
          CD        IF (J.NE.1) WRITE (6,110) J
          CD110     FORMAT (' RDLSI TEST-SRQ FAILURE: '14)
          C         IF (J.NE.1) GOTO 100
          C
          C         POLL FOR RESPONSE
0005                J=IBUP (6,IADR)
0006                IF (J.EQ."102) GOTO 200 !SUCCESS
0007                IF (J.EQ.-6) GOTO 230   !TIMEOUT
          D         WRITE (6,120) J
          D120      FORMAT (' RDLSI POLL FAILURE: '14)
0008                GOTO 100                !TRY AGAIN
          C
          C         READ THE BYTE STRING
0009      200       J=IBUP (1,IADR,IBUF,ILEN)
0010                IF (J.EQ.ILEN) ISW=1    !SUCCESS
0011                IF (J.NE.ILEN) ISW=J    !FAILURE
          D         WRITE (6,210) J,ISW
          D210      FORMAT (/' RDLSI STATUS: J='13' ISW='13)
          D         WRITE (6,220) IBUF
          D220      FORMAT (' RDLSI STRING='1204)
          C
0012                RETURN
          C
0013      230       CONTINUE
          D         WRITE (6,240) J
          D240      FORMAT (' RDLSI TIMEOUT: J='13)
0014                RETURN
          C
0015                END


PROGRAM SECTIONS

 Name        Size                    Attributes

$CODE1   000216    71          RW,I,CON,LCL
$PDATA   000010     4          RW,D,CON,LCL
$IDATA   000040    16          RW,D,CON,LCL
$VARS    000004     2          RW,D,CON,LCL


Total Space Allocated = 000272    93

No FPP Instructions Generated
```

```
FORTRAN IV-PLUS V3.0            21:07:58    25-Mar-82            Page 1
WRLSI.FTN;101           /TR:BLOCKS/WR

0001                SUBROUTINE WRLSI (IADR,STRING,ICNT,ISW)
                C
                C       SUBROUTINE TO WRITE A BYTE STRING TO THE LSI VOLTMETER.
                C
                C       ARGUMENTS:
                C           IADR   --  BUS ADDRESS OF DEVICE
                C           STRING--   BYTE ARRAY TO BE SENT
                C           ICNT   --  LENGTH OF ARRAY TO BE SENT
                C           ISW    --  STATUS OF OPERATION
                C                          ISW=1 IS SUCCESSFUL
                C                          ISW=* UNSUCCESSFUL (* = GPIB ERROR)
                C
                C       AUTHOR: R. E. SCOTT, JR. (NRL/USRD); VERSION: JANUARY 1982
                C
0002                BYTE STRING(ICNT)
                C
                C       WRITE THE STRING
0003                J=IBUP(0,IADR,STRING,ICNT)
                C
                C       DID IT WORK? (IF ISW = ICNT, SUCCESS).
                C       IF NOT, ISW IS GPIB ERROR STATUS.
0004                IF (J.NE.ICNT) ISW=J
0005                IF (J.EQ.ICNT) ISW=1
                C
                D       WRITE (6,90) STRING
                D90     FORMAT (' WRLSI STRING='(ICNT)A1)
                D       WRITE (6,100) IADR,J,ISW
                D100    FORMAT (' WRLSI STATUS: ADDR='O3' J='I3' ISW='I3)
                C
0006                RETURN
0007                END


PROGRAM SECTIONS

  Name        Size                    Attributes

$CODE1    000134     46               RW,I,CON,LCL
$PDATA    000004      2               RW,D,CON,LCL
$IDATA    000032     13               RW,D,CON,LCL
$VARS     000002      1               RW,D,CON,LCL


Total Space Allocated = 000174     62

No FPP Instructions Generated
```

```
FORTRAN IV-PLUS V3.0          21:08:16    25-Mar-82          Page 1
WRLPA.FTN;101          /TR:BLOCKS/WR

0001                SUBROUTINE WRLPA (IADR,ICODE,IVALUE,ISW)
            C
            C       SUBROUTINE TO WRITE A CODE AND AN INTEGER VALUE TO
            C       THE LSI-VOLTMETER, AND READ AN ACKNOWLEDGEMENT.
            C               IADR   = GPIB BUS ADDRESS OF VOLTMETER
            C               ICODE  = TWO-BYTE PARAMETER SETUP CODE
            C               IVALUE = POSITIVE INTEGER PARAMETER TO LOAD
            C               ISW    = STATUS OF OPERATION
            C                        ISW = 1 (SUCCESSFUL AS ACKNOWLEDGED)
            C                        ISW = * (FAILURE * AS ACKNOWLEDGED)
            C
            C       AUTHOR: R. E. SCOTT, JR. (NRL/USRD); VERSION: JANUARY 1982
            C
0002                BYTE STRING(8)
0003                BYTE ICODE(2)
            C
            C       INCORPORATE THE CODE INTO THE STRING
0004                STRING(1)=ICODE(1)
0005                STRING(2)=ICODE(2)
            C
            C       ENCODE PARAMETER INTO SIX-CHARACTER ASCII STRING
0006                ENCODE (6,100,STRING(3)) IVALUE
0007        100     FORMAT (I6)
            C
            C       REPLACE LEADING SPACES WITH ZEROS
0008                DO 110 I=3,8
0009        110     IF (STRING(I).EQ.' ') STRING(I)='0'
            C
0010                NB=8
            C
            C       WRITE THE STRING TO THE GPIB-BUS
            D       WRITE (6,200) IVALUE,STRING
            D200    FORMAT (' WRLPA VALUE='I6' STRING='8A1)
0011                CALL WRLSI (IADR,STRING,NB,ISW)
0012                IF (ISW.NE.1) CALL TSTERR (6,ISW)        !ERROR IS ???
            C       IF (ISW.EQ.-6) J=IBUP(10)                !TIMEOUT CLEANUP
            D       WRITE (6,300) ISW
            D300    FORMAT (' WRLPA (WRITE) STATUS: ISW='I3)
            C
            C       READ REPLY IF GIVEN
            C       CALL RDLSI (IADR,ISTAT,2,ISW)
            C       IF (ISW.NE.1) CALL TSTERR (6,ISW)        !ERROR IS ???
            C       IF (ISW.EQ.-6) J=IBUP(10)                !TIMEOUT CLEANUP
            CD      WRITE (6,310) ISW
            CD310   FORMAT (' WRLPA (READ) STATUS='I3)
            C
            C       SUBSTITUTE ACKNOWLEDGED STATUS FOR GPIB STATUS
            C       ISW=ISTAT
0013                RETURN
0014                END


PROGRAM SECTIONS

  Name        Size                  Attributes

$CODE1    000252    85              RW,I,CON,LCL
$PDATA    000010     4              RW,D,CON,LCL
$IDATA    000032    13              RW,D,CON,LCL
$VARS     000014     6              RW,D,CON,LCL


Total Space Allocated = 000330   108

No FPP Instructions Generated
```

```
FORTRAN IV-PLUS V3.0          21:08:36    25-Mar-82          Page 1
WRLVM.FTN;101          /TR:BLOCKS/WR

0001                SUBROUTINE WRLVM (IADR,ICODE,IVAL,ICNT,ISW)
            C
            C       SUBROUTINE TO WRITE A BYTE STRING TO THE LSI-VOLTMETER
            C       PREFIXED BY A TWO-BYTE PARAMETER CODE.
            C               IADR  = GPIB BUS ADDRESS
            C               ICODE = TWO-BYTE PARAMETER CODE
            C               IVAL  = ARRAY TO SEND TO VOLTMETER
            C               ICNT  = NUMBER OF INTEGERS TO SEND
            C               ISW   = STATUS OF ACKNOWLEDGEMENT
            C       IVAL COULD BE A 4-BYTE REAL VARIABLE (ICNT=2).
            C       IVAL COULD BE NOTHING (ICNT=0; SEND CODE ONLY).
            C
            C       AUTHOR: R. E. SCOTT, JR. (NRL/USRD); VERSION: JANUARY 1982
            C
0002                INTEGER ICODE
0003                INTEGER IVAL(8)
0004                INTEGER STRING(10)
            C
            C       PREFIX CODE INTO STRING
0005                STRING(1)=ICODE
0006                IF (ICNT.EQ.0) GOTO 60
            C
            C       ENTER VALUE INTO STRING, TWO BYTES AT A TIME
0007                DO 50 I=1,ICNT
0008        50      STRING(I+1)=IVAL(I)
            C
            C       COMPUTE STRING LENGTH
0009        60      NB=2+2*ICNT
            C
            C       WRITE THE STRING
0010                CALL WRLSI (IADR,STRING,NB,ISW)
0011                IF (ISW.NE.1) CALL TSTERR (6,ISW)          !ERROR IS ???
            C       IF (ISW.EQ.-6) J=IBUP(10)                  !TIMEOUT CLEANUP
            D       IF (ISW.NE.1) WRITE (6,100) ISW
            D100    FORMAT (' WRLVM STATUS: ISW='I3)
            C
            C       READ REPLY IF GIVEN
            C       CALL RDLSI (IADR,ISTAT,2,ISW)
            C       IF (ISW.NE.1) CALL TSTERR (6,ISW)          !ERROR IS ???
            C       IF (ISW.EQ.-6) J=IBUP(10)                  !TIMEOUT CLEANUP
            CD      IF (ISW.NE.1) WRITE (6,110) ISW
            CD110   FORMAT (' RDLSI STATUS: ISW='I3)
            C
            C       EXCHANGE GPIB STATUS FOR ACKNOWLEDGEMENT STATUS
            C       ISW=ISTAT
0012                RETURN
0013                END


PROGRAM SECTIONS

 Name       Size                  Attributes

$CODE1   000234    78             RW,I,CON,LCL
$PDATA   000004     2             RW,D,CON,LCL
$IDATA   000032    13             RW,D,CON,LCL
$VARS    000030    12             RW,D,CON,LCL
$TEMPS   000002     1             RW,D,CON,LCL


Total Space Allocated = 000324   106

No FPP Instructions Generated
```

```
FORTRAN IV-PLUS V3.0            21:09:27     25-Mar-82          Page 1
WRMEA.FTN;1                /TR:BLOCKS/WR

0001                SUBROUTINE WRMEA (IADR,ICODE,FREQ,ITIME,ISTAT)
          C
          C         SUBROUTINE TO WRITE A "ME" (MEASURE) TO THE LSI-VOLTMETER,
          C         (WAIT), ENABLE SAMPLING SYNTHESIZER, RECEIVE ACKNOWLEDGEMT,
          C         AND DISABLE SAMPLING SYNTHESIZER.
          C         *** SPECIFICALLY FOR LOW FREQUENCY SYSTEM SYNTHESIZER ***
          C
          C         IADR......IEEE-488 BUS ADDRESS FOR VOLTMETER.
          C         ICODE.....MEASUREMENT CODE (ME).
          C         FREQ......SAMPLING FREQUENCY IN HZ.
          C         ITIME.....TIME TO WAIT BETWEEN MEASURE COMMAND AND ENABLING
          C                   SAMPLE-RATE GENERATOR (MSEC).
          C         ISTAT.....MEASUREMENT STATUS.
          C
          C         AUTHOR: R. E. SCOTT, JR. (NRL/USRD); VERSION: JANUARY 1982
          C
0002                ISTAT=0                     !STATUS: NO MEASUREMENT
          C
          C         COMMAND A MEASUREMENT
0003                CALL WRLSI (IADR,'ME',2,ISW)
0004                IF(ISW.NE.1)CALL TSTERR (6,ISW) !BUS ERROR?
          D         IF (ISW.NE.0) WRITE (6,100) ISW
          D100      FORMAT (' WRMEA WRITE-"ME" STATUS: '13)
0005                IF (ISW.NE.0) RETURN              !GIVE UP IF BUS ERROR
          C
0006                CALL WAIT (ITIME,1,K)             !WAIT FOR SLAVE TO BE READY
0007                CALL SM102 ("24,FREQ,'E ',JSW,ISW)!ENABLE SYNTHESIZER (CLOCK)
0008                CALL RDLSI (IADR,ISTAT,2,ISW)     !GET STATUS FROM SLAVE
0009                IF(ISW.NE.1)CALL TSTERR (6,ISW) !BUS ERROR?
          D         IF (ISW.NE.1) WRITE (6,110) ISW
          D110      FORMAT (' WRMEA ACKNOWLEDGE STATUS: '13)
0010                CALL SM102 ("24,FREQ,'D ',JSW,ISW)!DISABLE SYNTHESIZER AGAIN
0011                RETURN
          C
0012                END


PROGRAM SECTIONS

 Name        Size                    Attributes

$CODE1    000252     85             RW,I,CON,LCL
$PDATA    000042     17             RW,D,CON,LCL
$IDATA    000064     26             RW,D,CON,LCL
$VARS     000006      3             RW,D,CON,LCL


Total Space Allocated = 000406   131

No FPP Instructions Generated
```

```
FORTRAN IV-PLUS V3.0          21:09:42    25-Mar-82          Page 1
METST.FTN;112          /TR:BLOCKS/WR

0001              SUBROUTINE METST (IAD,IT,MODE)

                  C       SUBROUTINE TO TEST THE LEESBURG/RD-11/LSI-11/A22D
                  C       HIGH-POWER MEASUREMENT SYSTEM INPUTS.

0002                      INCLUDE 'LSPARM.COM'
0003  *                   COMMON /LSPARM/SIFREQ,SIAMPL,SAFREQ,SAAMPL,
      *                   1IGAIN(3),NAD,NADT,NPAD,NPTR,NPST,NCYC,NHAR,
      *                   2IAVG,IDST,ICTYP,IAERS,IGATE,IWINDO,IDLUN,IHARC,
      *                   3IPORT(3)
      *     C             SIFREQ,SIAMPL-----SIGNAL FREQUENCY AND AMPLITUDE
      *     C             SAFREQ,SAAMPL-----SAMPLE FREQUENCY AND AMPLITUDE
      *     C             IGAIN-----A/D PROGRAMMABLE GAIN
      *     C             NAD-------WHICH A/D'S USED (1=1,2=2,3=4,ALL=7)
      *     C             NADT------WHICH A/D (DMA) TO INTERRUPT ON
      *     C             NPAD------NUMBER OF POINTS TO CONVERT
      *     C             NPTR------NUMBER OF POINTS TO TRANSFER
      *     C             NPST------FIRST POINT TO TRANSFER
      *     C             NCYC------NUMBER OF CYCLES TO CONVERT
      *     C             NHAR------NUMBER OF HARMONICS USED TO COMPUTE DISTORTION
      *     C             IAVG------NUMBER OF CYCLES TO AVERAGE TO (0=DON'T)
      *     C             IDST------DISPLAY STYLE (0--5)
      *     C             ICTYP-----COMPUTATION TYPE
      *     C             IAERS-----A/D ERRORS TO RETRY BEFORE QUITTING
      *     C             IGATE-----GATED (1) OR NOT (0) (DRV-11 IN USE))
      *     C             IWINDO----MULTIPLY DATA BY WINDOW (WDATA)
      *     C             IDLUN-----DISPLAY DEVICE
      *     C             IHARC-----HARMONIC TO COMPUTE
      *     C
0004                      INCLUDE 'LSDATA.COM'
0005  *                   PARAMETER BSIZE=1024                    !MAXIMUM POINTS TO READ
0006  *                   PARAMETER NOAD=3                        !NUMBER OF CHANNELS
0007  *                   COMMON /IDAT/IDATA(NOAD*BSIZE)          !RAW A/D DATA (IN BITS)
0008  *                   COMMON /RDAT/RDATA(NOAD*BSIZE)          !NORMALIZED DATA (IN VOLTS)
0009  *                   COMMON /WDAT/WDATA(BSIZE),WATT,WAV      !WINDOW, ATTENUATION, AVERAGE
0010  *                   COMMON /SDAT/SDATA(BSIZE)               !COMPUTATION BUFFER
0011  *                   COMMON /RTIM/TIM(NOAD)                  !COMPUTATION TIMES
      *     C
            C
0012                      INTEGER IPAR(15),IAE(6)

0013                      IF (IT.EQ.'ME') GOTO 100
0014                      IF (IT.EQ.'R1') GOTO 200
0015                      IF (IT.EQ.'R2') GOTO 200
0016                      IF (IT.EQ.'R3') GOTO 200
            C
            C             CODE TO BE USED TO PERFORM A/D CONVERSION (ME)
0017        100           CONTINUE
            C             WRITE (6,101)
            C101          FORMAT (' DO A/D CONVERSION')
0018                      IF (MODE.NE.'LO') CALL WRLVM (IAD,'ME',0,0,ISW)
0019                      IF (MODE.EQ.'LO') CALL WRMEA (IAD,'ME',SAFREQ,200,ISW)
0020                      IF (ISW.GE.0) RETURN
0021                      WRITE (6,102) ISW
0022        102           FORMAT (' ***  A/D CONVERSION ERROR: '13' ***')
0023                      IF (ISW.EQ.-100) WRITE (6,110)
0024                      IF (ISW.EQ.-101) WRITE (6,120)
0025        110           FORMAT (' A/D SETUP ERROR (#A/D'S OR INTERRUPT)')
0026        120           FORMAT (' TOO MANY POINTS REQUESTED')
0027                      RETURN
            C
            C             READ A DATA BUFFER FROM THE LSI (R1,R2,R3)
0028        200           CONTINUE
            C             WRITE (6,201)
            C201          FORMAT (' READ FROM 11/03')
0029                      RDS=SECNDS(0.0)
0030                      CALL WRLVM (IAD,IT,0,0,ISW)
```

```
0031                IF (ISW.NE.1) WRITE (6,203) ISW
0032      203       FORMAT (' WRLVM ERROR: '13)
0033                IF (ISW.EQ.-6) J=IBUP(10)
0034                IF (ISW.NE.1) RETURN
0035                IF (IT.EQ.'R1') J=1
0036                IF (IT.EQ.'R2') J=2
0037                IF (IT.EQ.'R3') J=3
0038                K=(J-1)*NPTR+1
0039                CALL RDLSI (IAD,IDATA(K),2*NPTR,ISW)
            C       WRITE (6,220) (IDATA(I),I=K,K+NPTR)
            C220    FORMAT (' '208)
0040                IF (ISW.NE.1) WRITE (6,230) J,ISW
0041      230       FORMAT (' *** A/D DATA CHANNEL '11' IN ERROR: '13' ***')
0042                IF (ISW.EQ.-6) J=IBUP(10)
0043                IF (ISW.NE.1) RETURN
            C
            C       FIX THE DATA PROPERLY
0044                L1=1+(J-1)*NPTR
0045                L2=L1+NPTR
0046                DO 250 I=L1,L2
0047                IM=IAND("7777,IDATA(I))
0048                IF (IM.GE."4000) IM=IOR("170000,IM)
0049                RDATA(I)=10.0*IM/"4000
0050      259       CONTINUE
0051                TIM(J)=SECNDS(RDS)
            D       IF ((IT.EQ.'R1').OR.(IT.EQ.'R2')) WRITE (6,260) TIM(J)
            D       IF ((IT.EQ.'R3').OR.(IT.EQ.'RE')) WRITE (6,260) TIM(J)
            D260    FORMAT (' BUFFER READ TIME='F8.3)
            C
0052                RETURN

0053                END
```

PROGRAM SECTIONS

Name	Size		Attributes
$CODE1	001156	311	RW,I,CON,LCL
$PDATA	000324	106	RW,D,CON,LCL
$IDATA	000056	23	RW,D,CON,LCL
$VARS	000074	30	RW,D,CON,LCL
LSPARM	000072	29	RW,D,OVR,GBL
IDAT	014000	3072	RW,D,OVR,GBL
RDAT	030000	6144	RW,D,OVR,GBL
WDAT	010010	2052	RW,D,OVR,GBL
XDAT	010000	2048	RW,D,OVR,GBL
ITIM	000014	6	RW,D,OVR,GBL

Total Space Allocated = 065772 13821

APPENDIX E

Voltmeter Program Listings

The modules used in this program are written in PDP-11 assembly language, divided into pure (ROM) code (modules 1-7) with all dynamic variables (RAM) in the final module.

(1) LSIVM — Command interpreter; measurement, computation, and display driver

(2) A32D — A/D converter control subroutine

(3) CONVRT — Floating to ASCII encoding subroutine, used to prepare numbers for display or transfer to the host

(4) DAPD — Computation of amplitude, phase, distortion; uses DGZL below to obtain real and imaginary parts of voltage

(5) DGZL — Single frequency DFT subroutine: computes real and imaginary parts of voltage of a given spectral line by Goertzel's algorithm

(6) GPIOSR — IEEE-488 bus input and output subroutines, initialize subroutine, and interrupt service routine

(7) GRMS — Computation by time-integration of rms, dc, positive peak, voltages

(8) LSIVAR — All the dynamic variables used by (1)-(6) above

Also used are modules from DEC's RSX-11M System Library--$ATAN (arctangent), $SIN (sine), and $SQRT (square root).

The subroutines (2)-(7) are written to conform with DEC standards; argument lists are passed through register 5 so that these routines could be used by other programs. Conditional assembly parameters allow dynamic variables to be assembled locally, or in an external global file such as LSIVAR.

(1) MODULE LSIVM - Main program for controlling the microcomputer-controlled sampling voltmeter

Internal Summary:

```
                        .TITLE  LSIVM
                        .IDENT  /06/
                        ;
                        ;
                        ;NRL/USRD MICROCOMPUTER SAMPLING VOLTMETER PROGRAM.
                        ;AUTHOR: RICHARD E. SCOTT, JR. (FEBRUARY 1982)
                        ;
                        ;
                        ;THIS PROGRAM IS DESIGNED TO PERFORM AS A SAMPLING VOLTMETER
                        ;CONTROLLED BY THE GPIB INTERFACE BUS. THE SETUP AND CONTROL
                        ;COMMANDS ARE SENT IN MULTI-BYTE PACKETS; DATA IS RETURNED
                        ;ON COMMAND IN THE FORM OF A WAVEFORM DIGITIZED BY UP TO THREE
                        ;A/D CONVERTERS, AND AS VARIOUS COMPUTED PARAMETERS.
                        ;
                        ;THE "MEASURE" COMMAND IS EFFECTIVELY THE ONLY ONE THAT IS
                        ;ACKNOWLEDGED BY THIS PROGRAM, RETURNING AN INTEGER STATUS.
                        ;
                        ;THE CURRENT VERSION CAN RETURN THE FOLLOWING PARAMETERS
                        ;UNDER HOST CONTROL: RMS, PEAK, AND DC VOLTAGE FROM TIME
                        ;INTEGRATION; RMS VOLTAGE, PHASE, AND DISTORTION FROM A DFT
                        ;CALCULATION. ALSO RETURNED IS TOTAL POWER (POWER IN ALL
                        ;HARMONICS).  PREPROCESSING OPTIONS INCLUDE MULTIPLYING THE
                        ;DATA SEQUENCE BY A WINDOW SPECIFIED BY THE HOST; AND
                        ;AVERAGING MANY-CYCLED SEQUENCES DOWN TO FEW CYCLES WHERE
                        ;SUCH IS PERMISSABLE.
                        ;
                        ;THE PROGRAM CONTAINS SEPARATE, CONTIGUOUS DATA BUFFERS OF A
                        ;GIVEN MAXIMUM SIZE FOR EACH OF THE THREE CHANNELS, HOWEVER,
                        ;DATA IS DIGITIZED INTO MEMORY AS IF THESE ARRAYS WERE CONTIG-
                        ;UOUS, BUT SIZED FOR THE NUMBER OF SAMPLES DIGITIZED.  THE
                        ;COMPUTATION IS PERFORMED ON ONE CHANNEL'S DATA AT A TIME, AS
                        ;EACH INTEGER SEQUENCE MUST FIRST BE CONVERTED TO FLOATING-POINT
                        ;VALUES STORED IN THE SINGLE "TEMPORARY" FLOATING ARRAY.  TOTAL
                        ;POWER, REQUIRING TWO FLOATING-POINT CHANNELS SIMULTANEOUSLY,
                        ;IS COMPUTED DIFFERENTLY (A SAMPLE AT A TIME).
                        ;
                        ;ALTHOUGH THIS DEVICE CAN ACT LIKE A DISPLAY VOLTMETER
                        ;INDEPENDENT OF COMPUTER CONTROL, IT IS STILL NECESSARY
                        ;TO SET UP THE DEVICE UNDER COMPUTER/BUS CONTROL; BECAUSE
                        ;IT IS FREQUENCY-DEPENDENT, NO DEFAULT VALUES ARE PRACTICAL
                        ;TO ALLOW IT TO POWER-ON INTO A VOLTAGE-READING MODE.
                        ;
                        ;THE VOLTMETER IS CONTROLLED BY COMMANDS FROM THE BUS,
                        ;WHICH CONSIST OF A TWO-BYTE (TWO CHARACTER) CODE, AND
                        ;AN INTEGER ARGUMENT (IF NEEDED) AS A BYTE STRING.
                        ;
                        ;ANY PROCESS CAN BE PERMANENTLY ABORTED FROM THE HOST
                        ;BY TRANSMITTING A DEVICE TRIGGER (GET) ON THE BUS.
                        ;ALL DEVICES AND COUNTERS ARE RESET.
```

```
                        ;SETUP COMMANDS (TWO BYTES PLUS BYTE STRING)-
                        ;FORMAT [TWO-BYTE CODE] [MULTI-BYTE VALUE]
                        ;    C#: PROGRAMMABLE GAIN FOR A/D CONVERTER #(1-3) (1,2,5,10)
                        ;    P#: INPUT PORT FOR A/D CONVERTER #(1-3) (0-16)
                        ;    AN: WHICH OF A/D'S USED (#1=1,#2=2,#3=4; ALL=7)
                        ;    AI: WHICH A/D TO INTERRUPT ON (DMA FINISH)
                        ;    AE: HOW MANY A/D ERRORS TO RETRY BEFORE GIVEUP
                        ;    PA: NUMBER OF POINTS (CONVERSIONS) PER WAVEFORM SEQUENCE
                        ;    PR: NUMBER OF POINTS (CONVERSIONS) TO RETURN TO HOST
                        ;    PS: FIRST POINT (CONVERSION) NUMBER TO RETURN TO HOST
                        ;    NC: NUMBER OF CYCLES PER SEQUENCE
                        ;    HA: NUMBER OF HARMONICS TO USE COMPUTING DISTORTION
                        ;    HC: WHICH HARMONIC TO COMPUTE
                        ;    DS: DISPLAY TYPE (Q.V.)
                        ;    DD: DISPLAY DEVICE (SEE PAGE 3)
                        ;    AV: AVERAGING MODE (0=NONE,N=CYCLES RESULTING)
                        ;    CT: COMPUTATION TYPE (Q.V.)
                        ;    GV: GATED (0=NO, 1=YES)
                        ;    WM: WINDOW MODE (0=NONE, 1=USE)
                        ;    MD: SEND MESSAGE TO DISPLAY
                        ;
                        ;
                        ;INFORMATIONAL COMMANDS (SIX BYTES)-(OBSOLETE)
                        ;FORMAT [TWO-BYTE CODE] [FOUR-BYTE REAL VALUE]
                        ;    FS: SIGNAL FREQUENCY IN HZ
                        ;    FC: CLOCK FREQUENCY IN HZ
                        ;
                        ;
                        ;CONTROL COMMANDS (TWO BYTES)-
                        ;FORMAT: [TWO-BYTE CODE]
                        ;    RS: CLEAR COUNTERS
                        ;    ME: PERFORM A/D CONVERSION AS PROGRAMMED
                        ;    LI: LIST 17 PARAMETERS ON THE LSIVM CONSOLE
                        ;    VM: GO TO VOLTMETER MODE (AUTOMATIC)
                        ;    WI: INPUT WINDOW DATA ("PR" REAL VALUES, 4*PR BYTES)
                        ;    TI: DUMMY COMMAND FOR I/O TIMING INFORMATION
                        ;    CW: COMPUTE SINE WAVEFORM FOR TIMING INFORMATION
                        ;
                        ;
                        ;DATA TRANSFER COMMANDS:
                        ;FORMAT:[TWO-BYTE CODE]
                        ;    R#: TRANSFER WAVEFORM # (1-3) SAMPLES BACK TO HOST
                        ;    RW: TRANSFER WINDOW AND AVERAGE BACK TO HOST
                        ;    RP: TRANSFER 18 PARAMETERS BACK TO HOST FOR CHECKING
                        ;ALSO.....
                        ;
                        ;
                        ;COMPUTE COMMANDS (TWO BYTES)-
                        ;FORMAT:[TWO-BYTE CODE]
                        ;    B#: RETURN VOLTAGE AND PHASE OF HCTH HARMONIC, CHANNEL #(1-3)
                        ;    C#: RETURN VOLTAGE AND PHASE OF FUNDAMENTAL, CHANNEL #(1-3)
                        ;    D#: RETURN VOLTAGE, PHASE, DISTORTION (FROM DFT), CHANNEL #(1-3)
                        ;    E#: RETURN RMS, PEAK+, PEAK-, DC VOLTAGES, CHANNEL #(1-3)
                        ;    F#: RETURN PEAK ONLY, CHANNEL #(1-3)
                        ;    PO: RETURN TOTAL POWER FROM CHANNEL #1 (VOLTS) AND #2 (AMPS)
                        ;    CJ: RETURN SECOND JOINT MOMENT AND COVARIANCE, CHANNELS #1 AND #2
```

```
                                ;SUBROUTINES REQUIRED:
                                ;
                                ;    CP10SR -- CP10-488 BUS CONTROL SUBROUTINES
                                ;    A32D   -- ADAC A/D CONVERTER CONTROL SUBROUTINE
                                ;    CONVRT -- REAL TO ASCII ENCODING FOR DISPLAY
                                ;    CRMS   -- COMPUTE TIME-SERIES VOLTAGES
                                ;    DGZL   -- COMPUTE [illegible] FOURIER TRANSFORM
                                ;    DAPD   -- COMPUTE [illegible] PHASE AND DISTORTION
                                ;    LSIVAR -- GLOBAL RAM VARIABLES
                                ;
                                ;
                                ;SPECIAL ASSEMBLY INSTRUCTIONS:
                                ;
                                ;THIS SYSTEM WAS DESIGNED TO BE LOADED INTO PROM, SO THE
                                ;RAM VARIABLES ARE CONCENTRATED IN THE MODULE LSIVAR.  IT
                                ;IS NECESSARY TO CHECK THE STATUS OF THE CONDITIONAL
                                ;ASSEMBLY PARAMETER "PROM" IN THE MAIN ROUTINE LSIVM AND
                                ;MANY OF THE SUBROUTINES TO BE SURE THAT IT IS IN THE RIGHT
                                ;STATE.  IN LSIVM PROM CONCERNS WHETHER PAGE ZERO (VECTOR
                                ;SPACE ) IS IN PROM (HOWEVER NOT); IN A32D PROM CONCERNS
                                ;WHETHER VECTORS ARE TO BE DYNAMICALLY LOADED; IN DGZL AND
                                ;DAPD IT CONCERNS WHETHER RAM VARIABLES ARE LOCALLY ASSEMBLED
                                ;(AS FOR OTHER USERS), OR IN LSIVAR (THIS APPLICATION).
                                ;
                                ;SPECIAL TASK-BUILD INSTRUCTIONS.  NOTE THAT THIS TASK IS
                                ;DESIGNED ONLY TO RUN ON A STAND-ALONE PDP-11, AND WILL NOT
                                ;EXECUTE UNDER RSX-11M.  IT IS NOT RELOCATABLE.
                                ;
                                ;TKB COMMAND FILE:
                                ;    LSIVM/SQ/-HM/-HD/FP,LS[illegible] G=SP:LSIVM
                                ;    CRMS,A32D,CONVRT,DGZL,[illegible]
                                ;    CP10SR
                                ;    LI:11SYSLIB.OLB/LB:SATAN:$SIN:$DSIN:$SQRT
                                ;    LSIVAR
                                ;    /
                                ;    STACK=0
                                ;    UNITS=9
                                ;    PAR=PAR:1000:1000
                                ;    ACTFIL=9
                                ;    MAXBUF=9
                                ;    CMLBUF=0
                                ;    /
                                ;
                                ;
                                ;
000000                          AC0=%0
000001                          AC1=%1
000002                          AC2=%2
```

```
                                        ;TEXT PRINTING ADDRESSES FOR SERIAL LINE INTERFACES
        177564                          DLVXSR=177564   ;CONSOLE (DLUN = 1)
        177566                          DLVXBF=177566
        176504                          CHXSR =176504   ;DISPLAY (DLUN = 2)
        176506                          CHXBF =176506

                                        ;GPIB REGISTERS REQUIRED
        160006                          GPACR=160006    ;GPIB AUXILIARY COMMAND REGISTER

                                        ;SET PROM = 1 IF VECTORS TO BE LOADED IN PROM.
                                        ;IF SO ALSO CHANGE TASKBUILD FILE TO PAR=PAR:6'...
        000001                          PROM=1


                                .IF NE PROM
000600  000167  000774                  JMP     START   ;LOCATION 0

000004                                  .BLKB   20      ;MISCELLANEOUS ERROR HALTS
000024  001000'                         .WORD   START   ;POWER-UP PC
000026  000340                          .WORD   340     ;POWER-UP PSW
000030                                  .BLKB   4
000034  002054'                         .WORD   FOISR
000036  000340                          .WORD   340

000040                                  .BLKB   70
                                        ;HARDWARE INTERRUPT VECTORS
000130  000000G                         .WORD   ADISR   ;A/D #1
000132  000340                          .WORD   340
000134                                  .BLKB   14
000150  000000G                         .WORD   DMISR   ;DMA #1
000152  000340                          .WORD   340

000154                                  .BLKB   14
000170  000000G                         .WORD   ADISR   ;A/D #2
000172  000340                          .WORD   340
000174                                  .BLKB   14
000210  000000G                         .WORD   DMISR   ;DMA #2
000212  000340                          .WORD   340

000214                                  .BLKB   14
000230  000000G                         .WORD   ADISR   ;A/D #3
000232  000340                          .WORD   340
000234                                  .BLKB   10
000244  002026'                         .WORD   FPISR   ;FLOATING-POINT
000246  000340                          .WORD   340
000250  000000G                         .WORD   DMISR   ;DMA #3
000252  000340                          .WORD   340

000254                                  .BLKB   14
000270  000000G                         .WORD   ISRCP   ;IEEE INTERFACE
000272  000340                          .WORD   340
000274                                  .BLKB   504
                                .ENDC
```

```
                                        ;START OF PROGRAM
                                        ;
                                        ;INITIALIZE WHATEVER IS NECESSARY AND GO INTO "WAIT" STATE.
                                        ;NORMAL STATUS IS TO WAIT FOR INPUT FROM THE IEEE-488 BUS;
                                        ;WHEN RECEIVED, GO TO EITHER DATA OR COMMAND INTERPRETER.
                                        ;
001006                          START:: .
001000  012706  000000G                 MOV     #STACK,SP       ;INITIALIZE STACK POINTER
001004  000005                          RESET                   ;CLEAR Q-BUS
                                        .IF EQ PROM
                                        MOV     #FPISR,@#244    ;LOAD FLOATING-ERROR INTERRUPT VECTOR
                                        MOV     #340,@#246      ;AND PSW
                                        MOV     #FOISR,@#34     ;LOAD FUNCTION-ERROR INTERRUPT VECTOR
                                        MOV     #340,@#36       ;AND PSW
                                        .ENDC
001006  005037  000000G                 CLR     @#ICA           ;NUMBER OF A/D CONVERSIONS
001012  005037  000000G                 CLR     @#ICE           ;NUMBER OF A/D ERRORS
001016  012703  001112'                 MOV     #CPINA,R3
001022  004767  000000G                 JSR     PC,CPINI        ;INITIALIZE IEEE-488 BUS CARD
001026  012700  000000                  MOV     #0,R0           ;CLEAR PSW
001032  106400                          MTPS    R0              ;(ENABLE INTERRUPTS)
                                        ;
                                        ;THIS PORTION OF THE PROGRAM AWAITS A DATA PACKET FROM THE
                                        ;HOST PROCESSOR AND DIRECTS IT TO THE PROPER FUNCTION.
                                        ;
001034                          RDCODE:
001034  012767  000001  000000G         MOV     #1,ISW          ;DEFAULT STATUS = 1
001042  012705  001102'                 MOV     #GPIARG,R5
001046  004767  000000G                 JSR     PC,GPIN         ;READ A CODE FROM THE GPIB
                                        ;
001052  023727  000000G 000002          CMP     @#IBLEN,#2      ;HOW MANY BYTES?
001060  001002                          BNE     10$
001062  000167  000772                  JMP     EXEC            ;IF TWO, EXECUTE A COMMAND
                                        ;OTHERWISE INTERPRET DATA
001066  022737  042115  000000G 10$:    CMP     #"MD,@#IC       ;CODE = "MD"?
001074  001012                          BNE     PARAM           ;NO
001076  000167  000626                  JMP     MESDAY          ;YES, REQUIRES SPECIAL PROCESSING
                                        ;
001102  000003  000000G 001116' GPIARG: 3,INBUF,MLEN,IBLEN
001110  000000G
001112  000001  002410'          GPINA:  1,ABORT
001116  000050                  MLEN:   .WORD   40.             ;MAXIMUM INPUT STRING LENGTH
001120  000000G                  BMAX::  .WORD   BSIZE           ;MAXIMUM DATA BUFFER SIZE
```

```
                                                ;THIS PORTION OF THE PROGRAM HANDLES THE INPUT OF DATA
                                                ;PARAMETERS.  RECEIVED BYTE STRING IS DECODED INTO AN
                                                ;INTEGER VARIABLE IN R3, AND THEN LOADED INTO THE VARIABLE
                                                ;SPECIFIED BY THE CODE WORD IC.

001122                                  PARAM:
001122  004767  000634                        JSR     PC,DECOD        ;DECODE ASCII STRING INTO BINARY INTEGER

001126  022737  030507  000000G               CMP     #"G1,@#IC       ;CODE = "G1"?
001134  001004                               BNE     10$
001136  010337  000000G                       MOV     R3,@#GAIN1      ;GAIN FOR CHANNEL 1
001142  000167  177666                       JMP     RDCODE

001146  022737  031107  000000G 10$:         CMP     #"G2,@#IC       ;CODE = "G2"?
001154  001004                               BNE     20$
001156  010337  000000G                       MOV     R3,@#GAIN2      ;GAIN FOR CHANNEL 2
001162  000167  177646                       JMP     RDCODE

001166  022737  031507  000000G 20$:         CMP     #"G3,@#IC       ;CODE = "G3"?
001174  001004                               BNE     30$
001176  010337  000000G                       MOV     R3,@#GAIN3      ;GAIN FOR CHANNEL 3
001202  000167  177626                       JMP     RDCODE

001206  022737  047101  000000G 30$:         CMP     #"AN,@#IC       ;CODE = "AN"?
001214  001004                               BNE     40$
001216  010337  000000G                       MOV     R3,@#ADS        ;NUMBER OF A/D'S IN USE
001222  000167  177606                       JMP     RDCODE

001226  022737  044501  000000G 40$:         CMP     #"AI,@#IC       ;CODE = "AI"?
001234  001004                               BNE     50$
001236  010337  000000G                       MOV     R3,@#ADT        ;WHICH A/D TO INTERRUPT ON (DMA FINISH)
001242  000167  177566                       JMP     RDCODE

001246  022737  040520  000000G 50$:         CMP     #"PA,@#IC       ;CODE = "PA"?
001254  001004                               BNE     60$
001256  010337  000000G                       MOV     R3,@#PAD        ;SAMPLES TO CONVERT PER DMA
001262  000167  177546                       JMP     RDCODE

001266  022737  051120  000000G 60$:         CMP     #"PR,@#IC       ;CODE = "PR"?
001274  001004                               BNE     70$
001276  010337  000000G                       MOV     R3,@#PTR        ;NUMBER SAMPLES TO RETURN TO HOST
001302  000167  177526                       JMP     RDCODE

001306  022737  051520  000000G 70$:         CMP     #"PS,@#IC       ;CODE = "PS"?
001314  001004                               BNE     80$
001316  010337  000000G                       MOV     R3,@#STPT       ;FIRST SAMPLE TO RETURN TO HOST
001322  000167  177506                       JMP     RDCODE

001326  022737  041516  000000G 80$:         CMP     #"NC,@#IC       ;CODE = "NC"?
001334  001004                               BNE     90$
001336  010337  000000G                       MOV     R3,@#NCYC       ;NUMBER CYCLES PER SEQUENCE
001342  000167  177466                       JMP     RDCODE

001346  022737  040510  000000G 90$:         CMP     #"HA,@#IC       ;CODE = "HA"?
001354  001003                               BNE     PARX
001356  010337  000000G                       MOV     R3,@#HAR        ;HOW MANY HARMONICS TO PROCESS
001362  000400                               BR      PARX
```

```
317 001364  022737  053161  000000G PARX:  CMP     #"AV,@#IC       ;CODE = "AV"?
318 001372  001004                         BNE     20$
319 001374  010337  000000G                MOV     R3,@#AVGCOD     ;AVERAGING PROCESS TO FOLLOW
320 001400  000167  177420                 JMP     RDCODE
321
322 001404  022737  051504  000000G 20$:   CMP     #"DS,@#IC       ;CODE = "DS"?
323 001412  001004                         BNE     30$
324 001414  010337  000000G                MOV     R3,@#DSTYLE     ;DISPLAY STYLE
325 001420  000167  177410                 JMP     RDCODE
326
327 001424  022737  053107  000000G 30$:   CMP     #"CV,@#IC       ;CODE = "CV"?
328 001432  001004                         BNE     40$
329 001434  010337  000000G                MOV     R3,@#GATED      ;GATING FLAG
330 001440  000167  177370                 JMP     RDCODE
331
332 001444  022737  042501  000000G 40$:   CMP     #"AE,@#IC       ;CODE = "AE"?
333 001452  001004                         BNE     50$
334 001454  010337  000000G                MOV     R3,@#AERRS      ;RETRY ERRORS COUNTER
335 001460  000167  177350                 JMP     RDCODE
336
337 001464  022737  052103  000000G 50$:   CMP     #"CT,@#IC       ;CODE = "CT"?
338 001472  001004                         BNE     60$
339 001474  010337  000000G                MOV     R3,@#COMTYP     ;COMPUTATION TYPE
340 001500  000167  177330                 JMP     RDCODE
341
342 001504  022737  042104  000000G 60$:   CMP     #"DD,@#IC       ;CODE = "DD"?
343 001512  001004                         BNE     70$
344 001514  010337  000000G                MOV     R3,@#DLUN       ;DISPLAY LUN
345 001520  000167  177310                 JMP     RDCODE
346
347 001524  022737  046527  000000G 70$:   CMP     #"WI,@#IC       ;CODE = "WM"?
348 001532  001004                         BNE     80$
349 001534  010337  000000G                MOV     R3,@#WMODE      ;WINDOW MODE
350 001540  000167  177270                 JMP     RDCODE
351
352 001544  022737  041510  000000G 80$:   CMP     #"HC,@#IC       ;CODE = "HC"?
353 001552  001004                         BNE     90$
354 001554  010337  000000G                MOV     R3,@#HTC        ;HARMONIC TO COMPUTE
355 001560  000167  177250                 JMP     RDCODE
356
357 001564  022737  030520  000000G 90$:   CMP     #"P1,@#IC       ;CODE = "P1"?
358 001572  001004                         BNE     100$
359 001574  010337  000000G                MOV     R3,@#AD1C       ;A/D CHANNEL 1 PORT
360 001600  000167  177230                 JMP     RDCODE
361
362 001604  022737  031120  000000G 100$:  CMP     #"P2,@#IC       ;CODE = "P2"?
363 001612  001004                         BNE     110$
364 001614  010337  000000G                MOV     R3,@#AD2C       ;A/D CHANNEL 2 PORT
365 001620  000167  177210                 JMP     RDCODE
366
367 001624  022737  031520  000000G 110$:  CMP     #"P3,@#IC       ;CODE = "P3"?
368 001632  001002                         BNE     120$
369 001634  010337  000000G                MOV     R3,@#AD3C       ;A/D CHANNEL 3 PORT
370 001640  000167  177170          120$:  JMP     RDCODE
371
```

```
                                        ;THIS PORTION WILL READ THE "REAL" PARAMETERS, SUCH AS
                                        ;SIGNAL AND SAMPLE FREQUENCY; THIS IS NO LONGER USED.
                                        ;DATA INPUT IS FOUR BYTES WHICH ARE MORE LITERALLY A
                                        ;32-BIT FLOATING VARIABLE IN PDP-11 FORMAT.

001644                          FREQP:
001644  022737  051506  000000C         CMP     #"FS,@#IC       ;IS THE CODE "FS"?
001652  001007                         BNE     10$
001654  013737  000000C 000000C         MOV     @#IP,@#FSIC     ;SIGNAL FREQUENCY
001662  013737  000002C 000002C         MOV     @#IP+2,@#FSIC+2
001670  000413                         BR      100$
001672  022737  041506  000000C 10$:    CMP     #"FC,@#IC       ;IS THE CODE "FC"?
001700  001007                         BNE     20$
001702  013737  000000C 000000C         MOV     @#IP,@#FSAM     ;SAMPLE FREQUENCY
001710  013737  000002C 000002C         MOV     @#IP+2,@#FSAM+2
001716  000402                         BR      100$

001720  000167  000400          20$:    JMP CERR                ;UNRECOGNIZED; ERROR!

001724  000167  177104          100$:   JMP RDCODE              ;RETURN FOR MORE INPUT


                                        ;ROUTINE FOR HANDLING "MESSAGE OF THE DAY"
                                        ;HOST MAY SEND UP TO 80-CHARACTER MESSAGE TO SLAVE TO
                                        ;BE DISPLAYED ON THE VOLTMETER

001730                          MESDAY:
001730  012705  001752'                 MOV     #MESARG,R5
001734  012737  006412  000000C         MOV     #6412,@#IC      ;REPLACE "MD" WITH CR-LF
001742  004767  006222                 JSR     PC,WRITE        ;OUTPUT STRING
001746  000167  177062                 JMP     RDCODE

001752  000002  000000C 000000C MESARG: 2,IC,!BLEN,BLUN
001760  000000C
```

```
                                        ;THIS SECTION WILL DECODE THE BYTE STRING FOLLOWING A
                                        ;SETUP COMMAND INTO AN INTEGER STORED IN R3.

002762                          DECOD:
002762  013700  000000C                 MOV     @#IBLEN,R0      ;LENGTH OF RECEIVED STRING
002766  162700  000002                  SUB     #2,R0           ;LESS CODE
002772  012701  000000?                 MOV     #IP,R1          ;ADDRESS OF RECEIVED STRING
002776  005003                          CLR     R3              ;CLEAR ACCUMULATED DATA WORD

003000  005002                  10$:    CLR     R2              ;TO BE SAFE
003002  112102                          MOVB    (R1)+,R2        ;GET AN ASCII BYTE
003004  162702  000060                  SUB     #60,R2          ;MAKE IT A BINARY DIGIT
003010  060203                          ADD     R2,R3           ;AND ADD TO DATA WORD
003012  005300                          DEC     R0              ;DONE YET?
003014  003403                          BLE     20$             ;FINISHED
003016  070327  000012                  MUL     #10.,R3         ;"SHIFT" LEFT
003022  000766                          BR      10$             ;NEXT DIGIT
003024  000207                  20$:    RTS     PC

                                        ;RUDIMENTARY HANDLING OF FLOATING-POINT EXCEPTIONS
                                        ;I. E. HALT ON ERROR; CLEAN UP ON CONTINUE (PROCEED)
003026                          FPISR::                         ;FLOATING-POINT INTERRUPT SERVICE
003026  010046                          MOV     R0,-(SP)
003030  010146                          MOV     R1,-(SP)
003032  170200                          STFPS   R0              ;STORE KEY-11A FPS
003034  170300                          STST    R0              ;STORE KEY-11A FEC
003036  000000                          HALT
003040  042700  100000                  BIC     #100000,R0      ;CLEAR ERROR BIT
003044  170100                          LDFPS   R0
003046  012601                          MOV     (SP)+,R1
003050  012600                          MOV     (SP)+,R0
003052  000002                          RTI
003054                          FOISR::                         ;ERROR INTERRUPTS FROM (FORTRAN) FUNCTIONS
003054  000000                          HALT
003056  000002                          RTI                     ;EG $ATAN2, $$SIN, $SQRT
```

```
                                                ;THIS PROGRAM INTERPRETS THE COMMAND CODES RECEIVED AND
                                                ;JUMPS TO THE APPROPRIATE ROUTINE.

002060                          EXEC:
002060  023727  000000G 051522         CMP     @#IC,#"RS       ;IS THE CODE "RS"?
002066  001002                        BNE     10$
002070  000167  000562                JMP     WAKE            ;"RESTART" THE CONVERSION COUNT

002074  023727  000000G 042515 10$:    CMP     @#IC,#"ME       ;IS THE CODE "ME"?
002102  001002                        BNE     20$
002104  000167  000402                JMP     MESURE          ;DO A/D CONVERSION

002110  023727  000000G 044514 20$:    CMP     @#IC,#"LI       ;IS THE CODE "LI"?
002116  001002                        BNE     30$
002120  000167  000552                JMP     PARLST          ;LIST PARAMETERS ON SLAVE

002124  023727  000000G 050122 30$:    CMP     @#IC,#"RP       ;IS THE CODE "RP"?
002132  001002                        BNE     40$
002134  000167  000764                JMP     PARSND          ;SEND PARAMETERS TO HOST

002140  023727  000000G 046526 40$:    CMP     @#IC,#"VM       ;IS THE CODE "VM"
002146  001002                        BNE     50$
002150  000167  003750                JMP     VMDISP          ;START VOLTMETER MODE

002154  123727  000000G 000122 50$:    CMPB    @#IC,#'R        ;IS THE CODE "R"#?
002162  001002                        BNE     70$
002164  000167  000760                JMP     ADSEND          ;SEND WAVEFORM TO HOST

002170  023727  000000G 044527 70$:    CMP     @#IC,#"WI       ;IS THE CODE "WI"?
002176  001002                        BNE     80$
002200  000167  002072                JMP     WINDIN          ;INPUT A WINDOW

002204  023727  000000G 044524 80$:    CMP     @#IC,#"TI       ;IS THE CODE TI?
002212  001002                        BNE     90$
002214  000167  176614                JMP     RDCODE

002220  023727  000000G 053503 90$:    CMP     @#IC,#"CW       ;IS THE CODE CW?
002226  001002                        BNE     100$
002230  000167  001460                JMP     CWAVE

                                                ;COMPUTATION MODES?
002234  123727  000000G 000102 100$:   CMPB    @#IC,#'B        ;IS THE CODE B#?
002242  001424                        BEQ     110$
002244  123727  000000G 000103        CMPB    @#IC,#'C        ;IS THE CODE C#? OR CJ?
002252  001420                        BEQ     110$
002254  123727  000000G 000104        CMPB    @#IC,#'D        ;IS THE CODE D#?
002262  001414                        BEQ     110$
002264  123727  000000G 000105        CMPB    @#IC,#'E        ;IS THE CODE E#?
002272  001410                        BEQ     110$
002274  123727  000000G 000106        CMPB    @#IC,#'F        ;IS THE CODE F#?
002302  001404                        BEQ     110$
002304  023727  000000G 047520        CMP     @#IC,#"PO       ;IS THE CODE PO?
002312  001002                        BNE     120$
002314  000167  001040        110$:   JMP     CMPUT           ;COMPUTE AND RETURN DATA TO HOST

002320  000167  000000        120$:   JMP     CERR            ;CODE NOT FOUND; ERROR
```

```
                                        ;THIS PORTION WILL TYPE OUT ERROR MESSAGES:
                                        ;CONDITION? UNRECOGNIZED CODE.
                                        
002324  012705  002360'          CERR:  MOV     #CERRA,R5        ;PRINT ERROR MESSAGE
002330  004767  005634                  JSR     PC,WRITE
002334  012705  002350'                  MOV     #CERRC,R5        ;PRINT ERRONEOUS CODE
002340  004767  003624                  JSR     PC,WRITE
                                        
002344  000167  176464                  JMP     RDCODE           ;RETURN FOR MORE INPUT
                                        
002350  000002  0000000G 0000000G CERRC: 2,IC,IBLEN,DLUN
002356  0000000G
                                        
002360  000002  002370' 002406' CERRA:  2,CERRT,CERRL,DLUN
002366  0000000G
002370     012      015      103  CERRT: .ASCII  <12><15>"CODE ERROR: "
002373     117      104      105
002376     040      105      122
002401     122      117      122
002404     072      040
                                        .EVEN
002406  000016                  CERRL:  .WORD   14.
                                        
                                        
                                        ;THIS PORTION WILL HANDLE IRREGULAR INTERRUPTIONS,
                                        ;IE, PROCESS ABORTS FROM THE HOST PROCESSOR.
                                        ;ACTIVATED BY DEVICE TRIGGER (GET) ON BUS.
                                        ;ENTERED FROM CPIOSR / INTERRUPT SERVICE ROUTINE
                                        
                                        
002410                          ABORT:
002410  012706  000000G                 MOV     #STACK,SP        ;INITIALIZE STACK
002414  012705  001112'                 MOV     #CPINA,R5
002420  004767  000000G                 JSR     PC,CPINI         ;INITIALIZE IEEE-488 BUS CARD
002424  012705  002446'                 MOV     #ABO,R5          ;PRINT MESSAGE
002430  004767  005534                  JSR     PC,WRITE
002434  012700  000000                  MOV     #0,R0            ;CLEAR PSW
002440  106400                          MTPS    R0               ;(ENABLE INTERRUPTS)
002442  000167  176366                  JMP     RDCODE           ;RETURN FOR INPUT
                                        
002446  000002  002456' 002510' ABO:    2,ABMES,ABLEN,DLUN
002454  0000000G
002456     015      052      052  ABMES: .ASCII  <15>"*** FUNCTION ABORTED ***"
002461     052      040      106
002464     125      116      103
002467     124      111      117
002472     116      040      101
002475     102      117      122
002500     124      105      104
002503     040      052      052
002506     052
                                        .EVEN
002510  000031                  ABLEN:  .WORD   25.
```

```
                                        ;THIS PORTION OF THE PROGRAM CONTROLS THE A/D CONVERSION
                                        ;AND DMA LOADING OF MEMORY WITH DATA.

                                        ;CHECK POINTS REQUESTED AGAINST BUFFER SIZE.
                                        ;TRY A/D CONVERSION AND REPEAT IF ERROR OCCURS.
                                        ;INCREMENT CONVERSION (AND ERROR?) COUNTER.

002512                          MEASURE:
002512  023737  001127' 000000'        CMP     @#BMAX,@#PTC    ;LIMIT BUFFER TO BMAX POINTS EACH CHANNEL
002520  003004                        BGT     10$             ;WITHIN LIMITS?
002522  112737  177677  000000G       MOVB    #-101,@#ISW     ;NO--PARAMETER ERROR (-101)
002530  000402                        BR      MACK            ;ACKNOWLEDGE THIS

002532  004767  000022          10$:  JSR     PC,DOAD         ;DO A/D CONVERSION

002536  012705  002552'        MACK: MOV     #CPACK,R5
002542  004767  000000G              JSR     PC,CPOUT        ;ACKNOWLEGEMENT
002546  000167  176262              JMP     RDCODE          ;NEXT COMMAND?
002552  000002  000000G 010164' CPACK: 2,ISW,TWO

002560  013737  000000G 000000G DOAD: MOV     @#AERRS,@#ECNT  ;ALLOW HOW MANY TRIES?
002566  012705  002636'        20$:  MOV     #ADCOM,R5
002572  004767  000000G              JSR     PC,A32D         ;EXECUTE A/D CONVERSION
002576  005237  000000G              INC     @#ICA           ;INCREMENT CONVERSION COUNTER
002602  005737  000000G              TST     @#ISW           ;ANY ERRORS IN A/D?
002606  002403                        BLT     30$
002610  004767  004122              JSR     PC,CHODIS
002614  000207                        RTS     PC              ;NO ERRORS!
002616  005237  000000G        30$:  INC     @#ICE           ;INCREMENT ERROR COUNTER
002622  004767  004110              JSR     PC,CHODIS       ;AND PRINT COUNT
002626  005337  000000G              DEC     @#ECNT          ;ERROR, SO TRY AGAIN
002632  003355                        BGT     20$             ;BUT ONLY AS ALLOWED
002634  000207                        RTS     PC

002636  000006  000000G 000000G 000000G ADCOM: 6,ADBUF,GAIN,PAD,ADS,GATED,ADIC,ISW
        002644  000000G 000000G 000000G
        002652  000000G 000000G

                                        ;THIS PORTION OF THE PROGRAM INITIALIZES THE TWO MAIN
                                        ;COUNTERS TO ZERO AND PRINTS THE "WAKE UP TEXT"

002656                          WAKE:
002656  005037  000000G              CLR     @#ICA           ;CLEAR A/D CONVERSION COUNTER
002662  005037  000000G              CLR     @#ICE           ;CLEAR A/D ERROR COUNTER
002666  004767  004044              JSR     PC,CHODIS       ;DISPLAY TEXT
002672  000167  176136              JMP     RDCODE
```

```
                                                ;THIS PORTION OF THE PROGRAM WILL LIST THE MAIN SETUP
                                                ;PARAMETERS ON THE LSIVM CONSOLE.

002676                                  PARLST:
002676  012737  000025  000000G                MOV     #21.,@#LCTR     ;NUMBER TO PRINT
002704  012737  000001  000000G                MOV     #1.,@#LPTR      ;STARTING AT PARAMETER 1
002712  012737  000000G 000000G                MOV     #DATA,@#LADD    ;AT ADDRESS "DATA"
002720  012705  003042'              10$:      MOV     #LST1,R5
002724  004767  005240                        JSR     PC,WRITE        ;PRINT INITIAL DESCRIPTION
002730  012705  003022'                        MOV     #LSTNO,R5
002734  004767  005320                        JSR     PC,PRINT        ;PRINT VARIABLE INDEX
002740  012705  003064'                        MOV     #LST2,R5
002744  004767  005220                        JSR     PC,WRITE        ;PRINT INTERNAL PUNCTUATION
002750  012705  003032'                        MOV     #LSTDA,R5
002754  017737  000000G 000000G                MOV     @LADD,@#LDAT
002762  004767  005272                        JSR     PC,PRINT        ;PRINT INTEGER VARIABLE
002766  005237  000000G                        INC     @#LADD
002772  005237  000000G                        INC     @#LADD
002776  005237  000000G                        INC     @#LPTR
003002  005337  000000G                        DEC     @#LCTR
003006  003403                                BLE     20$             ;ANOTHER ONE?
003010  004767  000066                        JSR     PC,IFWAIT       ;YES, WAIT TO SEE THIS ONE
003014  000741                                BR      10$

003016  000167  176012              20$:      JMP     RDCODE          ;RETURN FOR MORE INPUT

003022  000002  000000G 000002    LSTNO:    2,LPTR,2,DLUN
003030  000000G
003032  000002  000000G 000006    LSTDA:    2,LDAT,6,DLUN
003040  000000G
003042  000002  003052' 003062'   LST1:     2,LIST1,LSTL1,DLUN
003050  000000G
003052     015     012     104    LIST1:    .ASCII  <15><12>"DATA ("
003055     101     124     101
003060     040     050

                                            .EVEN
003062  000010                    LSTL1:    .WORD   LSTL1-LIST1

003064  000002  003074' 003100'   LST2:     2,LIST2,LSTL2,DLUN
003072  000000G
003074     051     040     075    LIST2:    .ASCII  ") = "
003077     040

                                            .EVEN
003100  000004                    LSTL2:    .WORD   LSTL2-LIST2

003102                              IFWAIT:                         ;DELAY ROUTINE (1 SEC??)
003102  012700  100000                        MOV     #100000,R0
003106  005300                      10$:      DEC     R0
003110  000240                                NOP
003112  000240                                NOP
003114  000240                                NOP
003116  000240                                NOP
003120  001372                                BNE     10$
003122  000207                                RTS     PC
```

```
                                                ;THIS PORTION OF THE PROGRAM IS DESIGNED TO SEND THE DATA
                                                ;PARAMETERS BACK TO THE HOST PROCESSOR FOR EXAMINATION.

003124                                  PARSND:
003124  012705  003140'                        MOV     #PARA,R5
003130  004767  000000G                        JSR     PC,CPOUT        ;OUTPUT DATA

003134  000167  175674                         JMP     RDCODE          ;RETURN FOR MORE INPUT
003140  000002  000000G 003146'  PARA:   2,DATA,PSIZE
003146  000052                   PSIZE:  .WORD   42.

                                                ;THIS PORTION OF THE PROGRAM IS DESIGNED TO SEND ONE OF THE
                                                ;THREE DATA BUFFERS BACK TO THE HOST PROCESSOR.  THE INPUT
                                                ;CODES ARE R1 (BUFFER 1), R2 (BUFFER 2), R3 (BUFFER 3).
                                                ;THE BUFFERS CONTAIN 12-BIT BINARY WORDS.

                                                ;ALSO TO TRANSFER BACK THE WINDOW (RW)--FLOATING WORDS.
                                                ;(PRECEDED BY COMPUTED AVERAGE).
003150                                  ADSEND:
003150  013737  003356'  000004G               MOV     @#RANA,@#RAA+4  ;LOAD ADDRESS INTO RAM
003156  023727  000000G  053522                CMP     @#IC,#"RW       ;WRITE WINDOW?
003164  001015                                 BNE     5$              ;NO
003166  013702  000000G                        MOV     @#PTR,R2        ;NUMBER OF FLOATING NUMBERS...
003172  005202                                 INC     R2              ;PLUS ONE FOR AVERAGE
003174  006302                                 ASL     R2              ;
003176  006302                                 ASL     R2              ;...NUMBER OF BYTES
003200  010237  000000G                        MOV     R2,@#ADWL
003204  012705  003350'                        MOV     #ADWA,R5
003210  004767  000000G                        JSR     PC,CPOUT        ;OUTPUTTED
003214  000167  175614                         JMP     RDCODE          ;RETURN FOR MORE INPUT

003220  012701  000000G          5$:    MOV     #ADBUF,R1       ;STORE ADDRESS
003224  023727  000000G  030522           CMP     @#IC,#"R1       ;WHAT CODE? (BUFFER?)
003232  001422                           BEQ     30$
003234  023727  000000G  031122           CMP     @#IC,#"R2
003242  001412                           BEQ     20$
003244  023727  000000G  031522           CMP     @#IC,#"R3
003252  001402                           BEQ     10$
003254  000167  177044                   JMP     CERR
003260  063701  000000G          10$:    ADD     @#PAD,R1        ;ADD IN OFFSETS FOR
003264  063701  000000G                  ADD     @#PAD,R1
003270  063701  000000G          20$:    ADD     @#PAD,R1        ;..FOR BUFFERS 2 AND 3
003274  063701  000000G                  ADD     @#PAD,R1
003300  063701  000000G          30$:    ADD     @#STPT,R1       ;AND FOR THE STARTING POINT
003304  063701  000000G                  ADD     @#STPT,R1       ;TWICE--WORD ADDRESS!
003310  010137  000002G                  MOV     R1,@#RAA+2

003314  013702  000000G                  MOV     @#PTR,R2        ;NUMBER OF POINTS TO TRANSFER
003320  060202                           ADD     R2,R2           ;DOUBLE FOR BYTES
003322  010237  000000G                  MOV     R2,@#RAN
003326  012737  000002  000000G          MOV     #2,@#RAA
003334  012705  000000G                  MOV     #RAA,R5
003340  004767  000000G                  JSR     PC,CPOUT        ;OUTPUT TO THE BUS

003344  000167  175464                   JMP     RDCODE          ;RETURN FOR MORE INPUT
003350  000002  000000G  000006G  ADWA:   2,WAV,ADWL
003356  000000G                  RANA:   RAN
```

```
                                        ;COMPUTATION "EXECUTIVE":
                                        ;1. CONVERT DATA TO REAL, SCALED NUMBERS
                                        ;2. MULTIPLY SEQUENCE BY WINDOW IF DESIRED
                                        ;3. AVERAGE SEQUENCE TO FEWER SAMPLES IF DESIRED
                                        ;4. REQUEST SPECIFIC COMPUTATION TYPE
                                        ;   A. DATA IS SCALED FOR GAIN IN A/D CONVERTER
                                        ;   B. DATA IS CORRECTED FOR WINDOW ATTENUATION IF NECESSARY.
                                        ;5  TRANSMIT COMPUTED DATA TO HOST.

003360                          COMPUT:
003360  023737  001120' 000000C        CMP     @#BMAX,@#PTR    ;LIMIT BUFFER TO BMAX POINTS EACH CHANNEL
003366  002003                        BGT     5$              ;WITHIN LIMITS?
003370  000000                        HALT
003372  000167  175436                JMP     RDCOPE          ;RETURN FOR MORE INPUT

003376  113706  000000C        5$:    MOVB    @#ICN,R6        ;GET CODE
003402  042700  177774                BIC     #177774,R0      ;CLEAR ALL BUT CHANNEL (1,2,3)
003406  005300                        DEC     R0              ;NOW (0,1,2)
003410  006100                        ROL     R0              ;NOW (0,2,4)
003412  006100                        ROL     R0              ;NOW (0,4,8)
003414  010037  000000C                MOV     R0,@#CHNO       ;USED FOR DISPLAYED DATA (OFFSET POINTER)

003420  013737  000000C 000000C        MOV     @#PTR,@#PTC     ;POINTS TO COMPUTE ON
003426  013737  000000C 000000C        MOV     @#NCYC,@#NCYF   ;CYCLES TO COMPUTE
003434  023727  000000C 047520        CMP     @#IC,#'PO       ;IF POWER, SKIP PREPROCESSING
003442  001510                        BEQ     90$
003444  023727  000000C 045103        CMP     @#IC,#'CJ       ;IF CO-VARIANCE, SKIP PREPROCESSING
003452  001512                        BEQ     100$
003454  004767  000442                JSR     PC,FLOT         ;FLOAT THE BUFFER
003460  123727  000000C 000106        CMPB    @#IC,#'F        ;SKIP PREPROCESSING FOR PEAK COMPUTATION
003466  001470                        BEQ     80$
003470  005737  000000C                TST     @#WMODE         ;MULTIPLY WINDOW?
003474  001402                        BEQ     10$             ;NO
003476  004767  000654                JSR     PC,WINDOW       ;YES
003502  005737  000000C        10$:    TST     @#AVGCOD        ;AVERAGE THE CYCLES?
003506  001402                        BEQ     20$             ;NO
003510  004767  000704                JSR     PC,BAVER        ;YES

003514  123727  000000C 000102  20$:    CMPB    @#IC,#'B
003522  001422                        BEQ     40$             ;COMPUTE DFT OF HARMONIC
003524  123727  000000C 000103        CMPB    @#IC,#'C
003532  001424                        BEQ     50$             ;COMPUTE VIA DFT (NO DISTORTION) (FUND.)
003534  123727  000000C 000104        CMPB    @#IC,#'D
003542  001426                        BEQ     60$             ;COMPUTE VIA DFT (INCLUDE DISTORTION)
003544  123727  000000C 000105        CMPB    @#IC,#'E
003552  001430                        BEQ     70$             ;COMPUTE VIA RMS (TIME AVERAGE)
003554  123727  000000C 000106        CMPB    @#IC,#'F
003562  001432                        BEQ     80$             ;COMPUTE VIA INSPECTION (PEAK ONLY)
003564  000167  176534                JMP     CERR            ;GIVE UP
```

```
747                                         ;COMPUTE AS SPECIFIED....
748                                         ;RETURN DATA SPECIFIED....
749
750 003570  004767  001146          40$:    JSR     PC,CSL          ;COMPUTE AMPLITUDE, PHASE OF HARMONIC
751 003574  004767  001256                  JSR     PC,TDFT         ;RETURN DATA TO HOST
752 003600  000167  175230                  JMP     RDCODE
753
754 003604  004767  001054          50$:    JSR     PC,CDFT         ;COMPUTE AMPLITUDE, PHASE OF FUNDAMENTAL
755 003610  004767  001242                  JSR     PC,TDFT         ;RETURN DATA TO HOST
756 003614  000167  175214                  JMP     RDCODE
757
758 003620  004767  000764          60$:    JSR     PC,CDFTD        ;COMPUTE AMPLITUDE, PHASE, DISTORTION
759 003624  004767  001204                  JSR     PC,TDFTD        ;RETURN DATA TO HOST
760 003630  000167  175200                  JMP     RDCODE
761
762 003634  004767  001342          70$:    JSR     PC,CRMSV        ;COMPUTE TIME-AVERAGED PARAMETERS
763 003640  004767  001514                  JSR     PC,TRMSV        ;RETURN DATA TO HOST
764 003644  000167  175164                  JMP     RDCODE
765
766 003650  004767  001420          80$:    JSR     PC,CPEKO        ;COMPUTE ABSOLUTE PEAK DATA
767 003654  004767  001544                  JSR     PC,TPEK         ;RETURN DATA TO HOST
768 003660  000167  175150                  JMP     RDCODE
769
770 003664  004767  001644          90$:    JSR     PC,CPWR         ;GET POWER IN CHANNELS #1-#2
771 003670  004767  002020                  JSR     PC,TPWR         ;RETURN DATA TO HOST
772 003674  000167  175134                  JMP     RDCODE
773
774 003700  004767  002030          100$:   JSR     PC,CJC          ;GET JOINT MOMENT AND COVARIANCE
775 003704  004767  002174                  JSR     PC,TJC          ;RETURN DATA TO THE HOST
776 003710  000167  175120                  JMP     RDCODE
```

```
                                        ;ROUTINE TO COMPUTE A SINUSOIDAL WAVEFORM FOR TESTING.
                                        ;V(I)=GAIN(N)*COS((I-1)*2*PI*NCYC/PTR)*2048./10.
                                        ;TEST ROUTINE == NOT NECESSARILY NEAT OR EFFICIENT!

003714                          CWAVE:
003714  012700  000000G                 MOV     #ADBUF,R0       ;START OF FIRST BUFFER
003720  063700  000000G                 ADD     @#STPT,R0       ;TRUE START OF FIRST WAVEFORM
003724  063700  000000G                 ADD     @#STPT,R0       ;(ADD STARTING POINT--TWICE FOR BYTES)
003730  063700  000000G                 ADD     @#PTR,R0        ;TRUE END OF FIRST WAVEFORM
003734  063700  000000G                 ADD     @#PTR,R0        ;(ALSO TWICE)
003740  013701  000000G                 MOV     @#PAD,R1        ;FIND END OF NEXT WAVEFORM
003744  063701  000000G                 ADD     @#PAD,R1        ;BY ADDING TWICE POINTS TO A/D
003750  060001                         ADD     R0,R1           ;TRUE END OF SECOND WAVEFORM
003752  013702  000000G                 MOV     @#PAD,R2        ;FIND END OF LAST WAVEFORM
003756  063702  000000G                 ADD     @#PAD,R2        ;BY ADDING TWICE POINTS TO A/D
003762  060102                         ADD     R1,R2           ;TRUE END OF THIRD WAVEFORM

003764  170001                         SETF
003766  177237  0000@9G                LDCIF   @#NCYC,AC2      ;NCYC FLOATED
003772  177137  000000G                LDCIF   @#PTR,AC1       ;PTR FLOATED
003776  174601                         DIVF    AC1,AC2         ;NCYC/PTR
004000  171237  004112'                MULF    @#TWOPI,AC2     ;2*PI*(NCYC/PTR)

004004  013703  000000G                 MOV     @#PTR,R3        ;POINTS TO COMPUTE
004010  177003                  10$:    LDCIF   R3,AC0          ;FLOAT I
004012  172067  000100                 ADDF    M1,AC0          ;(I-1)
004016  171002                         MULF    AC2,AC0         ;(I-1)*(--)
004020  010046                         MOV     R0,-(SP)        ;SAVE REGISTERS ON STACK
004022  010146                         MOV     R1,-(SP)
004024  010246                         MOV     R2,-(SP)
004026  010346                         MOV     R3,-(SP)
004030  174246                         STF     AC2,-(SP)
004032  004767  000000G                JSR     PC,$$COS        ;COS(2*PI*(I-1)*NCYC/PTR)
004036  174437  004272'                DIVF    @#FFAC,AC0      ;(--)*2048./10.
004042  172626                         LDF     (SP)+,AC2       ;RESTORE REGISTERS FROM STACK
004044  012603                         MOV     (SP)+,R3
004046  012602                         MOV     (SP)+,R2
004050  012601                         MOV     (SP)+,R1
004052  012600                         MOV     (SP)+,R0

004054  177137  000000G                LDCIF   @#GAIN1,AC1     ;FLOAT GAIN (SCALE)
004060  171100                         MULF    AC0,AC1         ;GAIN(I)*(--)
004062  175540                         STCFI   AC1,-(R0)       ;SIMULATED A/D CONVERTED VALUE

004064  177137  000000G                LDCIF   @#GAIN2,AC1     ;AGAIN
004070  171100                         MULF    AC0,AC1
004072  175541                         STCFI   AC1,-(R1)

004074  177137  000000G                LDCIF   @#GAIN3,AC1     ;AGAIN
004100  171100                         MULF    AC0,AC1
004102  175542                         STCFI   AC1,-(R2)

004104  077337                         SOB     R3,10$          ;NEXT SAMPLE (WORKING FROM END TO FRONT)
004106  000167  174722                 JMP     RDCODE
004112  040711  007733          TWOPI: .FLT2   6.2831854
004116  140200  000000          M1:    .FLT2   -1.
```

```
836                                             ;ROUTINE TO "FLOAT" THE A/D DATA BUFFER FOR PROCESSING
837                                             ;(CONVERT INTEGER TO FLOATING NUMBERS IN VOLTS IN RBUF).
838
839 004122                               FLGT:
840 004122  010046                              MOV     R0,-(SP)        ;SAVE REGISTERS ON STACK
841 004124  010146                              MOV     R1,-(SP)
842 004126  010246                              MOV     R2,-(SP)
843 004130  170001                              SETF
844
845 004132  012701  000000G                     MOV     #ADBUF,R1       ;STORE ADDRESS
846 004136  123727  000000G 000061              CMPB    @#ICN,#'1       ;STARTING WHERE?
847 004144  001422                              BEQ     30$
848 004146  123727  000000G 000062              CMPB    @#ICN,#'2
849 004154  001412                              BEQ     20$
850 004156  123727  000000G 000063              CMPB    @#ICN,#'3
851 004164  001402                              BEQ     10$
852 004166  000167  176132                      JMP     CERR
853 004172  063701  000000G             10$:    ADD     @#PAD,R1        ;ADD IN OFFSETS FOR
854 004176  063701  000000G                     ADD     @#PAD,R1
855 004202  063701  000000G             20$:    ADD     @#PAD,R1        ;..FOR BUFFERS 2 AND 3
856 004206  063701  000000G                     ADD     @#PAD,R1
857 004212  063701  000000G             30$:    ADD     @#STPT,R1       ;AND FOR THE STARTING POINT
858 004216  063701  000000G                     ADD     @#STPT,R1       ;TWICE FOR BYTES
859
860 004222  013700  000000G                     MOV     @#PTC,R0
861 004226  012702  000000G                     MOV     #RBUF,R2
862 004232  042711  170000             100$:   BIC     #170000,(R1)    ;CLEAR TO TWELVE BITS
863 004236  021127  004000                      CMP     (R1),#4000      ;IS IT NEGATIVE?
864 004242  002402                              BLT     110$            ;NO
865 004244  052711  170000                      BIS     #170000,(R1)    ;YES--SIGN EXTEND
866
867 004250  177021                      110$:   LDCIF   (R1)+,AC0       ;CONVERT TO FLOATING
868 004252  171037  004272'                     MULF    @#FFAC,AC0      ;DIVIDE BY HALF FULL SCALE (11 BITS)
869                                                                     ;MULTIPLY BY HALF FULL SCALE (10V)
870 004256  174022                              STF     AC0,(R2)+
871 004260  077014                              SOB     R0,100$
872
873 004262  012602                              MOV     (SP)+,R2        ;RESTORE REGISTERS FROM STACK
874 004264  012601                              MOV     (SP)+,R1
875 004266  012600                              MOV     (SP)+,R0
876 004270  000207                              RTS     PC
877
878 004272  036240  000000             FFAC:   .FLT2   .0048828125     ; = 10./2048.
```

```
                                        ;ROUTINE TO READ THE VALUES OF A SEQUENCE-WINDOW
                                        ;FROM THE HOST.
                                        ;NOTE THAT THE WINDOW MUST MATCH THE ENTIRE SEQUENCE
                                        ;LENGTH (PTR=PTC).

004276                          WINDIN:
004276  012765  004344'                 MOV     #CPWIN,R5
004302  004767  000000G                 JSR     PC,CPIN         ;GET DATA UNTIL TERMINATOR

                                        ;DATA THAT HAS A WINDOW APPLIED IS REDUCED IN MAGNITUDE
                                        ;BY THE AVERAGE OF THE WINDOW.  THIS AVERAGE MUST BE
                                        ;COMPUTED AND FACTORED INTO ALL RESULTS (VIA DEGAIN).

004306  170400                          CLRF    AC0             ;INITIALIZE SUM
004310  012700  000000G                 MOV     #WIND,R0        ;ADDRESS OF WINDOW DATA
004314  013701  000000G                 MOV     @#PTR,R1        ;POINTS TO AVERAGE
004320  170001                          SETF
004322  172020                  10$:    ADDF    (R0)+,AC0       ;SUM WINDOW
004324  077102                          SOB     R1,10$
004326  177137  000000G                 LDCIF   @#PTR,AC1       ;POINTS USED
004332  174401                          DIVF    AC1,AC0         ;AVERAGED VALUE
004334  174037  000000G                 STF     AC0,@#WAV       ;STORED

004340  000167  174470                  JMP     RDCODE
004344  000002  000000G 004354' CPWIN:  2,WIND,NWIN,ADWL
004352  000000G
004354  000000G                 NWIN:   .WORD   4*BSIZE


                                        ;SUBROUTINE TO MULTIPLY THE DATA BUFFER RBUF BY THE
                                        ;STORED WINDOW FUNCTION.

004356                          WINDOW:
004356  005737  000000G                 TST     @#WMODE         ;DESIRED?
004362  001001                          BNE     10$             ;YES
004364  000207                          RTS     PC

004366  012700  000000G         10$:    MOV     #RBUF,R0        ;BUFFER ADDRESS
004372  013701  000000G                 MOV     @#PTC,R1        ;POINTS TO BE WINDOWED
004376  012702  000000G                 MOV     #WIND,R2        ;WINDOW ADDRESS

004402  170001                          SETF
004404  172410                  20$:    LDF     (R0),AC0        ;DATA WORD
004406  172522                          LDF     (R2)+,AC1       ;WINDOW WORD
004410  171001                          MULF    AC1,AC0         ;PRODUCT
004412  174020                          STF     AC0,(R0)+       ;STORE BACK TO DATA ARRAY
004414  077105                          SOB     R1,20$          ;CONTINUE TILL DONE

004416  000207                          RTS     PC
```

```
                                        ;SUBROUTINE TO AVERAGE A SEQUENCE OF CYCLES---
                                        ;NOTE THAT IF THE NUMBER OF POINTS PER SUBSEQUENCE IS INTEGRAL,
                                        ;THE NCYC CYCLES CAN BE AVERAGED INTO NCYF CYCLE'S WORTH
                                        ;OF DATA, MAKING SUBSEQUENT COMPUTATION LESS TIME-CONSUMING.

004420                          BAVER:
004420  005737  000000G                TST     @#AVGCOD        ;ALLOWABLE?
004424  001001                        BNE     5$              ;YES
004426  000207                        RTS     PC
004430  022737  000001  000000G 5$:    CMP     #1,@#NCYC       ;ONLY ONE CYCLE?
004436  001001                        BNE     10$             ;NO
004440  000207                        RTS     PC              ;YES--FORGET IT!

004442  013701  000000G         10$:  MOV     @#NCYC,R1       ;NUMBER OF CYCLES PER SEQUENCE
004446  006700                        SXT     R0
004450  071037  000000G               DIV     @#AVGCOD,R0     ;NUMBER OF CYCLES PER SUBSEQUENCE
004454  005701                        TST     R1              ;COME OUT EVEN?
004456  001401                        BEQ     13$             ;OKAY
004460  000207                        RTS     PC              ;NO--CAN'T AVERAGE

004462  022700  000001          13$:  CMP     #1,R0           ;NUMBER OF SUBSEQUENCES TO AVERAGE
004466  001001                        BNE     14$             ;ONE?
004470  000207                        RTS     PC              ;YES--DON'T BOTHER
004472  010037  000000G         14$:  MOV     R0,@#NCAV       ;MORE--SAVE

004476  013701  000000G               MOV     @#PTC,R1        ;TOTAL POINTS IN SEQUENCE
004502  070137  000000G               MUL     @#AVGCOD,R1     ;..TIMES CYCLES IN SUBSEQUENCE
004506  006700                        SXT     R0
004510  071037  000000G               DIV     @#NCYC,R0       ;DIVIDED BY CYCLES IN SEQUENCE
004514  005701                        TST     R1              ;REMAINDER ZERO?
004516  001401                        BEQ     20$             ;YES; GO ON
004520  000207                        RTS     PC              ;NO--CAN'T AVERAGE
004522  013737  000000G 000000G 20$:  MOV     @#AVGCOD,@#NCYF ;CYCLES PER SUBSEQUENCE; NON-ZERO

                                        ;R0 = POINTS PER NEW SUBSEQUENCE
004530  010004                        MOV     R0,R4           ;R4 = POINTS PER SUBSEQUENCE (COUNTER)
004532  010037  000000G               MOV     R0,@#PTC        ;NEW POINTS TO CONVERT
004536  006300                        ASL     R0
004540  006300                        ASL     R0              ;R0 = ADDRESS OFFSET FOR 4-BYTE WORDS
004542  013701  000000G               MOV     @#NCAV,R1       ;R1 = NUMBER OF SUBSEQUENCES (COUNTER)
004546  012702  000000G               MOV     #RBUF,R2        ;R2 = DATA ADDRESS (INPUT)
004552  012703  000000G               MOV     #RBUF,R3        ;R3 = DATA ADDRESS (OUTPUT)
004556  170001                        SETF
004560  177101                        LDCIF   R1,AC1          ;AC1 = NUMBER OF SUBSEQUENCES

004562  170400                  25$:  CLRF    AC0             ;AC0 = RUNNING SUM

004564  172012                  30$:  ADDF    (R2),AC0        ;SUM VALUES FROM EACH CYCLE
004566  060002                        ADD     R0,R2           ;POINT TO NEXT CYCLE DATA WORD
004570  077103                        SOB     R1,30$          ;DONE YET?
004572  174401                        DIVF    AC1,AC0         ;YES--COMPUTE AVERAGE
004574  174023                        STF     AC0,(R3)+       ;SAVE AVERAGED POINT
004576  010302                        MOV     R3,R2           ;NEXT INPUT ADDRESS
004600  013701  000000G               MOV     @#NCAV,R1       ;RESTORE COUNTER
004604  077412                        SOB     R4,25$          ;NEXT POINT ON COUNTER

004606  000207                        RTS     PC
```

```
                                                ;COMPUTE THE VOLTAGES VIA DISCRETE FOURIER TRANSFORMS (DFT):
                                                ;TYPE "B"--VOLTAGE AND PHASE OF NTH HARMONIC;
                                                ;TYPE "C"--VOLTAGE AND PHASE OF FUNDAMENTAL;
                                                ;TYPE "D"--VOLTAGE, PHASE, AND DISTORTION OF FUNDAMENTAL.
                                                ;VOLTAGES ARE CORRECTED FOR A/D GAIN AND ARE RMS.
                                                ;DISTORTION IS IN PERCENTAGE.

004610                                  GDFTD:
004610  012705  004642'                        MOV     #AMAG,R5        ;COMPUTE THE DFT VALUES AND DISTORTION
004614  004767  000000G                        JSR     PC,DAPD         ;DGRTZL-TYPE
004620  172437  000000G                        LDF     @#VDFT,AC0      ;CORRECT FOR A/D GAIN
004624  004767  000640                        JSR     PC,DEGAIN
004630  004767  000662                        JSR     PC,DEWIND       ;AND FOR WINDOW IF USED
004634  174037  000000G                        STF     AC0,@#VDFT
004640  000207                                RTS     PC
004642  000007  000000G 000000G AMAG:    7,RBUF,PTC,NCYF,VDFT,PDFT,DIST,HAR,IERD
004650  000000G 000000G 000000G
004656  000000G 000000G 000000G

004664                                  CDFT:
004664  012705  004716'                        MOV     #ADFT,R5        ;COMPUTE THE DFT VALUES (NO DISTORTION)
004670  004767  000000G                        JSR     PC,DAPD         ;DGRTZL-TYPE
004674  172437  000000G                        LDF     @#VDFT,AC0      ;CORRECT FOR A/D GAIN
004700  004767  000564                        JSR     PC,DEGAIN
004704  004767  000606                        JSR     PC,DEWIND       ;AND FOR WINDOW IF USED
004710  174037  000000G                        STF     AC0,@#VDFT
004714  000207                                RTS     PC
004716  000007  000000G 000000G ADFT:    7,RBUF,PTC,NCYF,VDFT,PDFT,DIST,NONE,IERD
004724  000000G 000000G 000000G
004732  000000G 004740' 000000G
004740  000000                          NONE:   .WORD   0


004742                                  CSL:
004742  013705  000000G                        MOV     @#HTC,R5        ;HARMONIC TO COMPUTE
004746  005205                                INC     R5              ;LINE NUMBER (1 = FUNDAMENTAL)
004750  070537  000000G                        MUL     @#NCYF,R5       ;NUMBER OF CYCLES OF THIS HARMONIC
004754  010537  000000G                        MOV     R5,@#NCYHA

004760  012705  005012'                        MOV     #ASL,R5
004764  004767  000000G                        JSR     PC,DAPD         ;COMPUTE SFDFT
004770  172437  000000G                        LDF     @#VDFT,AC0      ;CORRECT FOR GAIN
004774  004767  000470                        JSR     PC,DEGAIN
005000  004767  000512                        JSR     PC,DEWIND       ;AND FOR WINDOW IF USED
005004  174037  000000G                        STF     AC0,@#VDFT
005010  000207                                RTS     PC

005012  000007  000000G 000000G ASL:     7,RBUF,PTC,NCYHA,VDFT,PDFT,DIST,NONE,IERD
005020  000000G 000000G 000000G
005026  000000G 004740' 000000G
```

```
                                                ;ROUTINE TO RETURN DATA TO THE HOST.
                                                ;ALSO TO SET UP ON-LINE DISPLAY VALUES.

005034                                  TDFTD:
005034  012705  005046'                         MOV     #TDFTA,R5       ;OUTPUT VOLTAGE, PHASE, DISTORTION
005040  004767  000000G                         JSR     PC,GPOUT
005044  000415                                  BR      DSLD
005046  000002  000000G 005054'        TDFTA:  2,DFD,N12
005054  000014                          N12:    .WORD   12.

005056                                  TDFT:
005056  012705  005070'                         MOV     #TDFA,R5        ;OUTPUT VOLTAGE, PHASE
005062  004767  000000G                         JSR     PC,GPOUT
005066  000413                                  BR      DSL
005070  000002  000000G 005076'        TDFA:   2,DFD,N8
005076  000010                          N8:     .WORD   8.

005100  172437  000000G                 DSLD:   LDF     @#DIST,AC0      ;LOAD DATA FOR DISPLAY ROUTINES
005104  012700  000000G                         MOV     #DIST1,R0       ;BASE ADDRESS FOR DISTORTIONS
005110  063700  000000G                         ADD     @#CHNO,R0       ;PARTICULAR ONE AVAILABLE
005114  174010                                  STF     AC0,(R0)        ;LOAD IT
005116  172437  000000G                 DSL:    LDF     @#VDFT,AC0      ;SAME FOR VOLTAGE
005122  012700  000000G                         MOV     #RMSV1,R0
005126  063700  000000G                         ADD     @#CHNO,R0
005132  174010                                  STF     AC0,(R0)
005134  172437  000000G                         LDF     @#PDFT,AC0      ;SAME FOR PHASE
005140  012700  000000G                         MOV     #PHAS1,R0
005144  063700  000000G                         ADD     @#CHNO,R0
005150  174010                                  STF     AC0,(R0)
005152  172437  000000G                         LDF     @#PHAS1,AC0     ;COMPUTE RELATIVE PHASE
005156  172537  000000G                         LDF     @#PHAS2,AC1
005162  173001                                  SUBF    AC1,AC0
005164  174037  000000G                         STF     AC0,@#RPHAS

005170  004767  001752                         JSR     PC,CH1DIS       ;POSSIBLE--ONE CHANNEL DISPLAY
005174  004767  002564                         JSR     PC,CH5DIS       ;POSSIBLE--TWO CHANNEL/RELATIVE PHASE
005200  000207                                  RTS     PC
```

```
                                                ;COMPUTE THE VOLTAGE PARAMETERS BY RMS TIME-AVERAGE:
                                                ;TYPE "E" -- GET RMS, PEAK, AND DC VOLTAGES.
                                                ;TYPE "F" -- GET ABS(PEAK) VOLTAGE ONLY.
                                                ;VOLTAGES ARE CORRECTED FOR A/D CONVERTER GAIN.

005202                                  GRMSV:
005202  012705  005342'                        MOV     #ARMSV,R5       ;CALL RMS COMPUTATION
005206  004767  000000G                        JSR     PC,GRMS
                                        
005212  172437  000000G                        LDF     @#RMSV,AC0      ;CORRECT FOR A/D GAIN (RMSV)
005216  004767  000246                        JSR     PC,DECAIN
005222  174037  000000G                        STF     AC0,@#RMSV
005226  172437  000000G                        LDF     @#DCV,AC0       ;CORRECT FOR A/D GAIN (DV)
005232  004767  000232                        JSR     PC,DECAIN
005236  174037  000000G                        STF     AC0,@#DCV
005242  172437  000000G                        LDF     @#PEKVP,AC0     ;CORRECT FOR A/D GAIN (MAXIMUM)
005246  004767  000216                        JSR     PC,DECAIN
005252  174037  000000G                        STF     AC0,@#PEKVP
005256  172437  000000G                        LDF     @#PEKVM,AC0     ;CORRECT FOR A/D GAIN (MINIMUM)
005262  004767  000202                        JSR     PC,DECAIN
005266  174037  000000G                        STF     AC0,@#PEKVM
005272  000207                                RTS     PC
                                        
005274                                  GPEKO:
005274  012701  000000G                        MOV     #RBUF,R1        ;FIND PEAK ONLY
005300  013700  000000G                        MOV     @#PTC,R0
005304  172421                                LDF     (R1)+,AC0       ;INITIALIZE PEAK
005306  170600                                ABSF    AC0             ;ABSOLUTE VALUE
005310  005300                                DEC     R0
005312  172521                          10$:  LDF     (R1)+,AC1       ;GET ANOTHER VALUE
005314  170601                                ABSF    AC1             ;ABSOLUTE VALUE
005316  173401                                CMPF    AC1,AC0         ;BIGGER?
005320  170000                                CFCC
005322  003401                                BLE     20$             ;NOPE
005324  172401                                LDF     AC1,AC0         ;YEP, SAVE NEW PEAK
005326  077007                          20$:  SOB     R0,10$          ;NEXT VALUE
005330  004767  000134                        JSR     PC,DECAIN       ;CORRECT FOR A/D GAIN
005334  174037  000000G                        STF     AC0,@#PEKVP
005340  000207                                RTS     PC
                                        
005342  000006  000000G 000000G 000000G ARMSV:  6,PTC,RBUF,RMSV,DCV,PEKVP,PEKVM
005350  000000G 000000G 000000G
005356  000000G
```

```
1113                                                    ;SUBROUTINES TO TRANSMIT RMS DATA TO THE HOST.
1114                                                    ;ALSO PREPARATION FOR REAL-TIME DISPLAYS.
1115
1116 005360                                    TRMSV:
1117 005360  012705  005414'                           MOV     #TRMSA,R5       ;TRANSFER THE DATA BACK
1118 005364  004767  000000G                           JSR     PC,CPOUT
1119 005370  172437  000000G                           LDF     @#RMSV,AC0      ;STORE FOR DISPLAYS
1120 005374  012700  000000G                           MOV     #RMSV1,R0       ;BASE ADDRESS OF RMS DATA
1121 005400  063700  000000G                           ADD     @#CHNO,R0       ;OFFSET FOR PARTICULAR CHANNEL
1122 005404  174010                                    STF     AC0,(R0)
1123 005406  004767  002154                           JSR     PC,CH4DIS
1124 005412  000207                                    RTS     PC
1125 005414  000002  000000G 005422'    TRMSA:  2,RMD,N16
1126 005422  000020                     N16:    .WORD   16.
1127
1128 005424                                    TPEK:
1129 005424  012705  005460'                           MOV     #TPEKA,R5       ;TRANSFER PEAK VOLTAGE ONLY
1130 005430  004767  000000G                           JSR     PC,CPOUT
1131 005434  172437  000000G                           LDF     @#PEKVP,AC0     ;LOAD FOR DISPLAY USE
1132 005440  012700  000000G                           MOV     #RMSV1,R0       ;BASE ADDRESS FOR "RMS" DATA
1133 005444  063700  000000G                           ADD     @#CHNO,R0       ;OFFSET FOR PARTICULAR CHANNEL
1134 005450  174010                                    STF     AC0,(R0)        ;STORE PEAK "AS RMS"
1135 005452  004767  002110                           JSR     PC,CH4DIS
1136 005456  000207                                    RTS     PC
1137 005460  000002  000000G 005466'    TPEKA:  2,PEKVP,N4
1138 005466  000004                     N4:     .WORD   4.
1139
1140
1141                                                    ;SUBROUTINE TO CONVERT THE COMPUTED VOLTAGES TO TRUE VALUES
1142                                                    ;BY REMOVING THE GAIN OF THE A/D CONVERTERS.
1143
1144                                                    ;ALSO TO CORRECT DATA COMPUTED WITH A KAISER WINDOW, BY
1145                                                    ;DIVIDING THE RESULTING MAGNITUDE BY THE WINDOW AVERAGE.
1146
1147 005470                                    DECAIN:                         ;SUBROUTINE TO REMOVE CHANNEL GAIN
1148 005470  113700  000000G                           MOVB    @#ICN,R0        ;EITHER "1", "2", OR "3"
1149 005474  042700  177774                           BIC     #177774,R0      ;EITHER 1, 2, OR 3
1150 005500  060000                                    ADD     R0,R0           ;OFFSET TIMES TWO
1151 005502  062700  177776G                           ADD     #GAIN-2,R0      ;BASE ADDRESS PLUS OFFSET FOR GAIN
1152 005506  170001                                    SETF
1153 005510  177110                                    LDCIF   (R0),AC1        ;GET GAIN
1154                                                                            ;VOLTAGE IN AC0 ALREADY
1155 005512  174401                                    DIVF    AC1,AC0         ;VDFT = VDFT / GAIN
1156 005514  000207                                    RTS     PC
1157
1158 005516                                    DEWIND:                         ;SUBROUTINE TO REMOVE WINDOW FACTOR
1159 005516  005737  000000G                           TST     @#WMODE         ;USING A WINDOW?
1160 005522  001403                                    BEQ     10$             ;NOT THIS TIME
1161 005524  172537  000000G                           LDF     @#WAV,AC1       ;GET AVERAGE (WAV)
1162 005530  174401                                    DIVF    AC1,AC0         ;VDFT = VDFT / WAV
1163 005532  000207                     10$:    RTS     PC
```

```
                                        ;SUBROUTINES TO COMPUTE THE TRUE (PSEUDO) POWER
                                        ;ASSUMING VOLTAGE AND CURRENT ARE IN CHANNELS #1-#2
                                        ;    PWR/K = SUM (V1(J)*V2(J)) / N     <FOR J=1,N>
                                        ;PSEUDO, FOR TRUE CURRENT IS UNKNOWN.  I(J) = K*V2(J).

005534                          GPWR:                                   ;COMPUTE PSEUDO-POWER
005534  010046                          MOV     R0,-(SP)                ;SAVE REGISTERS ON STACK
005536  010146                          MOV     R1,-(SP)
005540  010246                          MOV     R2,-(SP)
005542  170001                          SETF
005544  012701  000000G                MOV     #ADBUF,R1               ;STORE ADDRESS OF VOLTAGE
005550  063701  000000G                ADD     @#STPT,R1               ;INCLUDING STARTING POINT
005554  063701  000000G                ADD     @#STPT,R1               ;TIMES TWO (INTEGER ADDRESS)
005560  010102                          MOV     R1,R2                   ;STORE ADDRESS OF CURRENT
005562  063702  000000G                ADD     @#PAD,R2                ;OFFSET TO SECOND BUFFER
005566  063702  000000G                ADD     @#PAD,R2                ;ALREADY INCLUDES STARTING POINT OFFSET
005572  013700  000000G                MOV     @#PTR,R0                ;AND POINTS TO COMPUTE

005576  170402                          CLRF    AC2                     ;INITIALIZE SUM

                                                                        ;FLOAT VOLTAGE....
005600  042711  170000          100$:   BIC     #170000,(R1)            ;CLEAR TO TWELVE BITS
005604  021127  004000                  CMP     (R1),#4000              ;IS IT NEGATIVE?
005610  002402                          BLT     110$                    ;NO
005612  052711  170000                  BIS     #170000,(R1)            ;YES--SIGN EXTEND
005616  177021                  110$:   LDCIF   (R1)+,AC0               ;CONVERT TO FLOATING
005620  171037  004272'                 MULF    @#FFAC,AC0              ;DIVIDE BY HALF FULL SCALE (11 BITS)
                                                                        ;MULTIPLY BY HALF FULL SCALE (10V)
005624  177137  000000G                LDCIF   @#CAIN1,AC1             ;GET GAIN OF VOLTAGE CHANNEL
005630  174401                          DIVF    AC1,AC0                 ;CONVERT TO TRUE VOLTS

                                                                        ;FLOAT CURRENT....
005632  042712  170000          120$:   BIC     #170000,(R2)            ;CLEAR TO TWELVE BITS
005636  021227  004000                  CMP     (R2),#4000              ;IS IT NEGATIVE?
005642  002402                          BLT     130$                    ;NO
005644  052712  170000                  BIS     #170000,(R2)            ;YES--SIGN EXTEND
005650  177122                  130$:   LDCIF   (R2)+,AC1               ;CONVERT TO FLOATING
005652  171137  004272'                 MULF    @#FFAC,AC1              ;DIVIDE BY HALF FULL SCALE (11 BITS)
                                                                        ;MULTIPLY BY HALF FULL SCALE (10V)
005656  171001                          MULF    AC1,AC0                 ;E(I)*I(I)
005660  177137  000000G                LDCIF   @#CAIN2,AC1             ;GET GAIN OF CURRENT CHANNEL
005664  174401                          DIVF    AC1,AC0                 ;CORRECT CURRENT FOR GAIN
005666  172200                          ADDF    AC0,AC2                 ;SUM PRODUCTS
005670  077035                          SOB     R0,100$                 ;NEXT?

005672  177037  000000G                LDCIF   @#PTR,AC0               ;POINTS DONE
005676  174600                          DIVF    AC0,AC2                 ;TAKE AVERAGE
005700  174237  000000G                STF     AC2,@#POWER             ;AND SAVE
005704  012602                          MOV     (SP)+,R2                ;RESTORE REGISTERS FROM STACK
005706  012601                          MOV     (SP)+,R1
005710  012600                          MOV     (SP)+,R0
005712  000207                          RTS     PC

005714  012705  005726'          TPWR:   MOV     #TPWRA,R5               ;RETURN PSEUDO-POWER
005720  004767  000000G                JSR     PC,CPOUT
005724  000207                          RTS     PC
005726  000002  000000G 005466' TPWRA:  2,POWER,N4
```

```
1223                                                ;SUBROUTINE TO COMPUTE THE SECOND JOINT MOMENT AND THE
1224                                                ;COVARIANCE (ANALAGOUS TO TOTAL POWER AND AC POWER).
1225
1226 005734                                 GJC:
1227 005734  004767  177574                        JSR     PC,GPWR         ;GET SECOND JOINT MOMENT ("POWER")
1228 005740  172437  000000G                       LDF     @#POWER,AC0
1229 005744  174037  000000G                       STF     AC0,@#SJM
1230
1231 005750  112737  000061  000000G               MOVB    #'1,@#ICN       ;FOR CHANNEL 1
1232 005756  004767  000050                        JSR     PC,CMN          ;GET MEAN VALUE
1233 005762  172437  000000G                       LDF     @#DCV,AC0
1234 005766  174037  000000G                       STF     AC0,@#COV
1235 005772  112737  000062  000000G               MOVB    #'2,@#ICN       ;FOR CHANNEL 2
1236 006000  004767  000026                        JSR     PC,CMN          ;GET MEAN VALUE
1237 006004  172437  000000G                       LDF     @#DCV,AC0
1238 006010  172537  000000G                       LDF     @#COV,AC1
1239 006014  171100                                MULF    AC0,AC1         ;"DC POWER" IN AC1
1240 006016  172437  000000G                       LDF     @#SJM,AC0
1241 006022  173001                                SUBF    AC1,AC0
1242 006024  174037  000000G                       STF     AC0,@#COV       ;SAVE COVARIANCE
1243 006030  000207                                RTS     PC
1244
1245 006032                                 CMN:                            ;FLOAT/AVERAGE/WINDOW/GET RMSV
1246 006032  013737  000000G 000000G               MOV     @#PTR,@#PTC     ;POINTS TO COMPUTE ON
1247 006040  013737  000000G 000000G               MOV     @#NCYC,@#NCYF   ;CYCLES TO COMPUTE
1248 006046  004767  176050                        JSR     PC,FLOT         ;FLOAT THE BUFFER
1249 006052  005737  000000G                       TST     @#WMODE         ;MULTIPLY WINDOW?
1250 006056  001402                                BEQ     10$             ;NO
1251 006060  004767  176272                        JSR     PC,WINDOW       ;YES
1252 006064  005737  000000G            10$:     TST     @#AVGCOD        ;AVERAGE THE CYCLES?
1253 006070  001402                                BEQ     20$             ;NO
1254 006072  004767  176322                        JSR     PC,BAVER        ;YES
1255 006076  004767  177100            20$:     JSR     PC,CRMSV        ;COMPUTE RMS DATA
1256
1257 006102  000207                                RTS     PC
1258
1259 006104  012705  006116'           TJC:     MOV     #TJCA,R5        ;RETURN DATA TO HOST
1260 006110  004767  000000G                       JSR     PC,CPOUT
1261 006114  000207                                RTS     PC
1262 006116  000002  000000G 005076'    TJCA:    2,SJM,N8
```

```
1264                                                    ;VOLTMETER SIMULATION ROUTINE.
1265                                                    ;IN THIS STATE THE SLAVE PROCESSOR WILL PRETEND TO BE A
1266                                                    ;SAMPLING VOLTMETER AND MEASURE/COMPUTE/DISPLAY AS FAST
1267                                                    ;AS IT CAN.  PRIOR TO THIS MODE BEING ENTERED, VARIOUS
1268                                                    ;PARAMETERS NEED TO BE SET UP (AS IN THE NORMAL COMPUTER-
1269                                                    ;CONTROLLED SITUATION).  THIS MODE PROCEEDS INDEPENDENT
1270                                                    ;OF ANY HOST CONTROL, ALTHOUGH THE HOST CAN ABORT THE MODE
1271                                                    ;AND THE SLAVE WILL REVERT TO THE INPUT-AWAITING MODE.
1272
1273 006124                                  VMDISP:
1274 006124  013737  000000G 000000G                  MOV     @#PTR,@#PTC     ;POINTS TO COMPUTE ON
1275 006132  013737  000000G 000000G                  MOV     @#NCYC,@#NCYF   ;CYCLES TO COMPUTE
1276 006140  004767  174414                           JSR     PC,DOAD         ;DO MEASUREMENT
1277
1278 006144  023727  000000G 000000                  CMP     @#DSTYLE,#0     ;NONE
1279 006152  001427                                  BEQ     40$
1280 006154  023727  000000G 000001                  CMP     @#DSTYLE,#1     ;CHANNEL 1
1281 006162  001424                                  BEQ     50$             ;COMPUTE VIA DFT
1282 006164  023727  000000G 000002                  CMP     @#DSTYLE,#2     ;CHANNEL 2
1283 006172  001420                                  BEQ     50$             ;COMPUTE VIA DFT
1284 006174  023727  000000G 000003                  CMP     @#DSTYLE,#3     ;CHANNEL 3
1285 006202  001414                                  BEQ     50$             ;COMPUTE VIA DFT
1286 006204  023727  000000G 000004                  CMP     @#DSTYLE,#4     ;VOLTAGES
1287 006212  001466                                  BEQ     80$             ;COMPUTE VIA RMS
1288 006214  023727  000000G 000005                  CMP     @#DSTYLE,#5     ;TWO-CHANNEL AND PHASE
1289 006222  001556                                  BEQ     90$
1290 006224  000000                                  HALT                    ;DISPLAY STYLE ERROR
1291 006226  000167  172602                           JMP     RDCODE
1292
1293 006232  000734                        40$:      BR      VMDISP          ;NO DISPLAY (ALREADY DONE)
1294
1295                                                    ;DISPLAY MODES 1,2,3 (SINGLE CHANNEL)
1296 006234  013700  000000G               50$:      MOV     @#DSTYLE,R0     ;STYLE = CHANNEL NUMBER
1297 006240  010037  000000G                         MOV     R0,@#CHNO       ;LOAD AS (1,2,3)
1298 006244  005337  000000G                         DEC     @#CHNO          ;BUT CONVERT TO (0,1,2)
1299 006250  006137  000000G                         ROL     @#CHNO          ;THEN (0,2,4)
1300 006254  006137  000000G                         ROL     @#CHNO          ;THEN (0,4,8)
1301 006260  062700  000060                          ADD     #60,R0          ;CONVERT (1-3) TO ASCII TO FOOL
1302 006264  110037  000000G                         MOVB    R0,@#ICN        ;INPUT MODE TO EXPECT CHANNEL N
1303 006270  004767  175626                          JSR     PC,FLOT         ;FLOAT DATA
1304 006274  004767  176056                          JSR     PC,WINDOW       ;MULTIPLY BY WINDOW
1305 006300  004767  176114                          JSR     PC,BAVER        ;AVERAGE DATA
1306 006304  004767  176300                          JSR     PC,CDFTD        ;COMPUTE DATA
1307 006310  172437  000000G                         LDF     @#VDFT,AC0      ;ACCESS VOLTAGE
1308 006314  012700  000000G                         MOV     #RMSV1,R0
1309 006320  063700  000000G                         ADD     @#CHNO,R0
1310 006324  174010                                  STF     AC0,(R0)
1311 006326  172437  000000G                         LDF     @#PDFT,AC0      ;ACCESS PHASE
1312 006332  012700  000000G                         MOV     #PHAS1,R0
1313 006336  063700  000000G                         ADD     @#CHNO,R0
1314 006342  174010                                  STF     AC0,(R0)
1315 006344  172437  000000G                         LDF     @#DIST,AC0      ;ACCESS DISTORTION
1316 006350  012700  000000G                         MOV     #DIST1,R0
1317 006354  063700  000000G                         ADD     @#CHNO,R0
1318 006360  174010                                  STF     AC0,(R0)
1319 006362  004767  000560                          JSR     PC,CH1DIS
1320 006366  000561                                  BR      100$
```

```
1322                                                            ;DISPLAY MODE 4 (THREE CHANNELS)
1323 006370  112737  000061  000000G 80$:   MOVB    #'1,@#ICN       ;CHANNEL 1
1324 006376  004767  175520                JSR     PC,FLOT         ;FLOAT DATA
1325 006402  004767  175750                JSR     PC,WINDOW       ;MULTIPLY BY WINDOW
1326 006406  004767  176006                JSR     PC,BAVER        ;AVERAGE DATA
1327 006412  004767  176564                JSR     PC,GRMSV        ;GET VOLTAGE
1328 006416  172437  000000G               LDF     @#RMSV,AC0      ;ACCESS VOLTAGE
1329 006422  174037  000000G               STF     AC0,@#RMSV1
1330 006426  112737  000062  000000G       MOVB    #'2,@#ICN       ;CHANNEL 2
1331 006434  013737  000000G 000000G       MOV     @#PTR,@#PTC     ;POINTS TO COMPUTE ON
1332 006442  013737  000000G 000000G       MOV     @#NCYC,@#NCYF   ;CYCLES TO COMPUTE
1333 006450  004767  175446                JSR     PC,FLOT         ;FLOAT DATA
1334 006454  004767  175676                JSR     PC,WINDOW       ;MULTIPLY BY WINDOW
1335 006460  004767  175734                JSR     PC,BAVER        ;AVERAGE DATA
1336 006464  004767  176512                JSR     PC,GRMSV        ;GET VOLTAGE
1337 006470  172437  000000G               LDF     @#RMSV,AC0      ;ACCESS VOLTAGE
1338 006474  174037  000000G               STF     AC0,@#RMSV2
1339 006500  112737  000063  000000G       MOVB    #'3,@#ICN       ;CHANNEL 3
1340 006506  013737  000000G 000000G       MOV     @#PTR,@#PTC     ;POINTS TO COMPUTE ON
1341 006514  013737  000000G 000000G       MOV     @#NCYC,@#NCYF   ;CYCLES TO COMPUTE
1342 006522  004767  175374                JSR     PC,FLOT         ;FLOAT DATA
1343 006526  004767  175624                JSR     PC,WINDOW       ;MULTIPLY BY WINDOW
1344 006532  004767  175662                JSR     PC,BAVER        ;AVERAGE DATA
1345 006536  004767  176440                JSR     PC,GRMSV        ;GET VOLTAGE
1346 006542  172437  000000G               LDF     @#RMSV,AC0      ;ACCESS VOLTAGE
1347 006546  174037  000000G               STF     AC0,@#RMSV3
1348 006552  004767  001010                JSR     PC,CH4DIS       ;DISPLAY DATA
1349 006556  000465                       BR      100$
1350
1351                                                            ;DISPLAY MODE 5 (DUAL CHANNEL)
1352 006560  112737  000061  000000G 90$:   MOVB    #'1,@#ICN
1353 006566  004767  175330                JSR     PC,FLOT         ;FLOAT DATA
1354 006572  004767  175560                JSR     PC,WINDOW       ;MULTIPLY BY WINDOW
1355 006576  004767  175616                JSR     PC,BAVER        ;AVERAGE DATA
1356 006602  004767  176056                JSR     PC,GDFT         ;GET VOLTAGE AND PHASE
1357 006606  172437  000000G               LDF     @#VDFT,AC0      ;ACCESS VOLTAGE
1358 006612  174037  000000G               STF     AC0,@#RMSV1
1359 006616  172437  000000G               LDF     @#PDFT,AC0      ;ACCESS PHASE
1360 006622  174037  000000G               STF     AC0,@#PHAS1
1361 006626  112737  000062  000000G       MOVB    #'2,@#ICN       ;CHANNEL 2
1362 006634  013737  000000G 000000G       MOV     @#PTR,@#PTC     ;POINTS TO COMPUTE ON
1363 006642  013737  000000G 000000G       MOV     @#NCYC,@#NCYF   ;CYCLES TO COMPUTE
1364 006650  004767  175246                JSR     PC,FLOT
1365 006654  004767  175476                JSR     PC,WINDOW       ;MULTIPLY BY WINDOW
1366 006660  004767  175534                JSR     PC,BAVER        ;AVERAGE DATA
1367 006664  004767  175774                JSR     PC,GDFT         ;GET VOLTAGE AND PHASE
1368 006670  172437  000000G               LDF     @#VDFT,AC0      ;ACCESS VOLTAGE
1369 006674  174037  000000G               STF     AC0,@#RMSV2
1370 006700  172437  000000G               LDF     @#PDFT,AC0      ;ACCESS PHASE
1371 006704  174037  000000G               STF     AC0,@#PHAS2
1372 006710  172437  000000G               LDF     @#PHAS1,AC0     ;COMPUTE RELATIVE PHASE
1373 006714  172537  000000G               LDF     @#PHAS2,AC1
1374 006720  173001                       SUBF    AC1,AC0
1375 006722  174037  000000G               STF     AC0,@#RPHAS
1376 006726  004767  001032                JSR     PC,CH5DIS       ;DISPLAY DATA
1377
1378 006732  000167  177166          100$:  JMP     VMDISP          ;AND AGAIN
```

```
                                                ;DISPLAY A/D CONVERSION COUNT AND ERROR COUNT,
                                                ;(IFF DSTYLE.EQ.0):
                                                ;FORMAT: A/D CONV=***** ERR=*****

006736  005737  000000G          CH0DIS: TST     @#DSTYLE        ;RIGHT DISPLAY STYLE?
006742  001401                          BEQ     10$             ;YES
006744  000207                          RTS     PC              ;NO
006746  012700  000000G          10$:    MOV     #CHERRY,R0      ;LOAD DISPLAY DELIMITERS
006752  012701  007114'                  MOV     #CH0BLK,R1

006756  112120                  20$:    MOVB    (R1)+,(R0)+     ;INTO RAM BUFFER
006760  001376                          BNE     20$

006762  013701  000000G                  MOV     @#ICA,R1        ;ENCODE CONVERSION COUNT...
006766  006700                          SXT     R0
006770  012702  000005                  MOV     #5,R2
006774  012703  000017G                  MOV     #CHERRY+15.,R3
007000  071027  000012          30$:    DIV     #10.,R0
007004  062701  000060                  ADD     #60,R1          ;...INTO A BYTE STRING...
007010  110143                          MOVB    R1,-(R3)
007012  010001                          MOV     R0,R1
007014  006700                          SXT     R0
007016  077210                          SOB     R2,30$

007020  013701  000000G                  MOV     @#ICE,R1        ;ENCODE ERROR COUNT...
007024  006700                          SXT     R0
007026  012702  000005                  MOV     #5,R2
007032  012703  000031G                  MOV     #CHERRY+25.,R3
007036  071027  000012          40$:    DIV     #10.,R0
007042  062701  000060                  ADD     #60,R1          ;...INTO A BYTE STRING...
007046  110143                          MOVB    R1,-(R3)
007050  010001                          MOV     R0,R1
007052  006700                          SXT     R0
007054  077210                          SOB     R2,40$

007056  012705  007070'                  MOV     #CH0ER,R5       ;...AND OUTPUT ENTIRE STRING.
007062  004767  001102          50$:    JSR     PC,WRITE
007066  000207                          RTS     PC

007070  000002  000000G 007100' CH0ER:  2,CHERRY,CH0LA,DLUN
007076  000000G
007100  000031                  CH0LA:  .WORD   25.
007102  000002  000000G 007112' CH0ES:  2,CHERRY,CH0LB,DLUN
007110  000000G
007112  000017                  CH0LB:  .WORD   15.
007114     015     101     057  CH0BLK: .ASCIZ  <15>"A/D CONV=    0 ERR=    0"
007117     104     040     103
007122     117     116     126
007125     075     040     040
007130     040     040     060
007133     040     105     122
007136     122     075     040
007141     040     040     040
007144     060     000

                                        .EVEN
```

```
1427                                                    ;SUBROUTINE TO DISPLAY DATA ON VOLTMETER
1428                                                    ;FORMAT #(1-3)= #)V=+**.** P:+***.* D:*.**
1429
1430 007146                                  CH1DIS:
1431 007146                                  CH2DIS:
1432 007146                                  CH3DIS:
1433 007146  012700  000000G                          MOV     #CHERRY,R0      ;LOAD DISPLAY DELIMITERS
1434 007152  012701  007532'                          MOV     #CH1BLK,R1
1435 007156  112120                          5$:     MOVB    (R1)+,(R0)+     ;INTO RAM BUFFER
1436 007160  001376                                  BNE     5$
1437
1438 007162  023727  000000G 000001                  CMP     @#DSTYLE,#1     ;RIGHT DISPLAY STYLE? (CHANNEL ONE)
1439 007170  001020                                  BNE     10$             ;YES
1440 007172  012705  007366'                          MOV     #CH1A,R5        ;CONVERT CHANNEL ONE VOLTAGE TO ASCII
1441 007176  004767  000000G                          JSR     PC,CONVRT
1442 007202  012705  007400'                          MOV     #CH1B,R5        ;CONVERT CHANNEL ONE PHASE TO ASCII
1443 007206  004767  000000G                          JSR     PC,CONVRT
1444 007212  012705  007412'                          MOV     #CH1C,R5        ;CONVERT CHANNEL ONE DISTORTION TO ASCII
1445 007216  004767  000000G                          JSR     PC,CONVRT
1446 007222  112737  000061  000001G                  MOVB    #'1,@#CHERRY+1
1447 007230  000451                                  BR      40$
1448
1449 007232  023727  000000G 000002  10$:     CMP     @#DSTYLE,#2     ;? (CHANNEL 2)
1450 007240  001020                                  BNE     20$
1451 007242  012705  007424'                          MOV     #CH2A,R5        ;CONVERT CHANNEL TWO VOLTAGE TO ASCII
1452 007246  004767  000000G                          JSR     PC,CONVRT
1453 007252  012705  007436'                          MOV     #CH2B,R5        ;CONVERT CHANNEL TWO PHASE TO ASCII
1454 007256  004767  000000G                          JSR     PC,CONVRT
1455 007262  012705  007450'                          MOV     #CH2C,R5        ;CONVERT CHANNEL TWO DISTORTION TO ASCII
1456 007266  004767  000000G                          JSR     PC,CONVRT
1457 007272  112737  000062  000001G                  MOVB    #'2,@#CHERRY+1
1458 007300  000425                                  BR      40$
1459
1460 007302  023727  000000G 000003  20$:     CMP     @#DSTYLE,#3     ;? (CHANNEL THREE)
1461 007310  001020                                  BNE     30$
1462 007312  012705  007462'                          MOV     #CH3A,R5        ;CONVERT CHANNEL THREE VOLTAGE TO ASCII
1463 007316  004767  000000G                          JSR     PC,CONVRT
1464 007322  012705  007474'                          MOV     #CH3B,R5        ;CONVERT CHANNEL THREE PHASE TO ASCII
1465 007326  004767  000000G                          JSR     PC,CONVRT
1466 007332  012705  007506'                          MOV     #CH3C,R5        ;CONVERT CHANNEL THREE DISTORTION TO ASCII
1467 007336  004767  000000G                          JSR     PC,CONVRT
1468 007342  112737  000063  000001G                  MOVB    #'3,@#CHERRY+1
1469 007350  000401                                  BR      40$
1470
1471 007352  000207                          30$:    RTS     PC              ;NO
1472
1473 007354  012705  007520'                  40$:    MOV     #CH1ER,R5       ;VALUE FILL DISPLAY
1474 007360  004767  000604                          JSR     PC,WRITE
1475 007364  000207                                  RTS     PC
1476
```

```
1478 007366  000004  000000G 000005G CH1A:   4,RMSV1,CHERRY+5.,TWO,TWO
     007374  010164' 010164'
1479 007400  000004  000000G 000016G CH1B:   4,PHAS1,CHERRY+14.,THREE,ONE
     007406  010166' 010162'
1480 007412  000004  000000G 000027G CH1C:   4,DIST1,CHERRY+23.,ONE,TWO
     007420  010162' 010164'
1481 007424  000004  000000G 000005G CH2A:   4,RMSV2,CHERRY+5.,TWO,TWO
     007432  010164' 010164'
1482 007436  000004  000000G 000016G CH2B:   4,PHAS2,CHERRY+14.,THREE,ONE
     007444  010166' 010162'
1483 007450  000004  000000G 000027G CH2C:   4,DIST2,CHERRY+23.,ONE,TWO
     007456  010162' 010164'
1484 007462  000004  000000G 000005G CH3A:   4,RMSV3,CHERRY+5.,TWO,TWO
     007470  010164' 010164'
1485 007474  000004  000000G 000016G CH3B:   4,PHAS3,CHERRY+14.,THREE,ONE
     007502  010166' 010162'
1486 007506  000004  000000G 000027G CH3C:   4,DIST3,CHERRY+23.,ONE,TWO
     007514  010162' 010164'
1487 007520  000002  000000G 007530' CH1ER:  2,CHERRY,CH1LA,DLUN
     007526  000000G
1488 007530  000034                 CH1LA:   .WORD   28.
1489
1490 007532     015     061     075 CH1BLK:  .ASCIZ  <15>"1=V=+##.## P=+###.# D=#.##"
     007535     126     075     053
     007540     043     043     056
     007543     043     043     040
     007546     120     075     053
     007551     043     043     043
     007554     056     043     040
     007557     104     075     043
     007562     056     043     043
     007565     000
1491                                 .EVEN
```

```
                                                ;SUBROUTINE TO DISPLAY DATA ON VOLTMETER
                                                ;FORMAT 4: V1=+*.** V2=+*.** V3=+*.***
                                
007566                                  CH4DIS:
007566  023727  000000C 000004                  CMP     @#DSTYLE,#4     ;RIGHT DISPLAY STYLE?
007574  001401                                  BEQ     10$             ;YES
007576  000207                                  RTS     PC              ;NO
007600  012700  000000C                 10$:    MOV     #CHERRY,R0      ;LOAD DISPLAY DELIMITERS
007604  012701  007726'                         MOV     #CH4BLK,R1
007610  112120                         20$:    MOVB    (R1)+,(R0)+     ;INTO RAM BUFFER
007612  001376                                  BNE     20$
007614  012705  007656'                         MOV     #CH4A,R5        ;CONVERT CHANNEL ONE VOLTAGE TO ASCII
007620  004767  000000C                         JSR     PC,CONVRT
007624  012705  007670'                         MOV     #CH4B,R5        ;CONVERT CHANNEL TWO VOLTAGE TO ASCII
007630  004767  000000C                         JSR     PC,CONVRT
007634  012705  007702'                         MOV     #CH4C,R5        ;CONVERT CHANNEL THREE VOLTAGE TO ASCII
007640  004767  000000C                         JSR     PC,CONVRT
007644  012705  007714'                         MOV     #CH4ER,R5       ;VALUE FILL DISPLAY
007650  004767  000314                         JSR     PC,WRITE
007654  000207                                  RTS     PC
                                
007656  000004  000000C 000004C        CH4A:   4,RMSV1,CHERRY+4.,ONE,TWO
007664  010162' 010164'
007670  000004  000000C 000015C        CH4B:   4,RMSV2,CHERRY+13.,ONE,TWO
007676  010162' 010164'
007702  000004  000000C 000026C        CH4C:   4,RMSV3,CHERRY+22.,ONE,THREE
007710  010162' 010166'
007714  000002  000000C 007724'        CH4ER:  2,CHERRY,CH4LA,DLUN
007722  000000C
007724  000034                         CH4LA:  .WORD   28.
                                
007726     015     126     061         CH4BLK: .ASCIZ  <15>"V1=+#.## V2=+#.## V3=+#.###"
007731     075     053     043
007734     056     043     043
007737     040     126     062
007742     075     053     043
007745     056     043     043
007750     040     126     063
007753     075     053     043
007756     056     043     043
007761     043     000
                                                .EVEN
```

```
1523                                                    ;SUBROUTINE TO DISPLAY DATA ON VOLTMETER
1524                                                    ;FORMAT 5: V1=+#.### V2=+#.### P=+##.#
1525
1526 007764                                    CH5DIS:
1527 007764  023727  000000G 000005                     CMP     @#DSTYLE,#5     ;RIGHT DISPLAY STYLE?
1528 007772  001401                                     BEQ     10$
1529 007774  000207                                     RTS     PC
1530 007776  012700  000000G                 10$:       MOV     #CHERRY,R0      ;LOAD DISPLAY DELIMITERS
1531 010002  012701  010124'                            MOV     #CH5BLK,R1
1532 010006  112120                         20$:       MOVB    (R1)+,(R0)+
1533 010010  001376                                     BNE     20$
1534 010012  012705  010054'                            MOV     #CH5A,R5        ;CONVERT CHANNEL ONE VOLTAGE TO ASCII
1535 010016  004767  000000G                            JSR     PC,CONVRT
1536 010022  012705  010066'                            MOV     #CH5B,R5        ;CONVERT CHANNEL TWO VOLTAGE TO ASCII
1537 010026  004767  000000G                            JSR     PC,CONVRT
1538 010032  012705  010100'                            MOV     #CH5C,R5        ;CONVERT RELATIVE PHASE TO ASCII
1539 010036  004767  000000G                            JSR     PC,CONVRT
1540 010042  012705  010112'                            MOV     #CH5ER,R5       ;VALUE FILL DISPLAY
1541 010046  004767  000116                             JSR     PC,WRITE
1542 010052  000207                                     RTS     PC
1543
1544 010054  000004  000000G 000004G CH5A:     4,RMSV1,CHERRY+4.,ONE,THREE
     010062  010162' 010166'
1545 010066  000004  000000G 000016G CH5B:     4,RMSV2,CHERRY+14.,ONE,THREE
     010074  010162' 010166'
1546 010100  000004  000000G 000027G CH5C:     4,RPHAS,CHERRY+23.,TWO,ONE
     010106  010164' 010162'
1547 010112  000002  000000G 010122' CH5ER:    2,CHERRY,CH5LA,DLUN
     010120  000000G
1548 010122  000034                    CH5LA:    .WORD   28.
1549
1550 010124     015     126     061   CH5BLK:   .ASCIZ  <15>"V1=+#.### V2=+#.### P=+##.#"
     010127     075     053     043
     010132     056     043     043
     010135     043     040     126
     010140     062     075     053
     010143     043     056     043
     010146     043     043     040
     010151     120     075     053
     010154     043     043     056
     010157     043     000
1551                                                    .EVEN
1552
1553 010162  000001                    ONE:      .WORD   1
1554 010164  000002                    TWO:      .WORD   2
1555 010166  000003                    THREE:    .WORD   3
```

```
1557                                                ;SUBROUTINES TO HANDLE CHARACTER OUTPUT:
1558                                                ;CALL WRITE (IBUFF,ILEN,DLUN)---ASCII STRING
1559                                                ;CALL PRINT (NUMBER,ILEN,DLUN)--DECIMAL NUMBER
1560
1561 010170  016500  000002          WRITE:  MOV     2(R5),R0        ;SET UP START ADDRESS
1562 010174  017501  000004                  MOV     @4(R5),R1       ;SET UP LENGTH
1563 010200  003404                          BLE     30$             ;IF NEG OR ZERO, ERROR
1564 010202  112002                  10$:    MOVB    (R0)+,R2        ;GET NEXT CHARACTER
1565 010204  004767  000004                  JSR     PC,PUTCHR
1566 010210  077104                          SOB     R1,10$          ;LOOP ON CHAR COUNT
1567 010212  000207                  30$:    RTS     PC
1568
1569 010214  022775  000002  000006  PUTCHR: CMP     #2,@6(R5)       ;WHICH DEVICE?
1570 010222  001407                          BEQ     CHCHR           ;CHERRY DISPLAY
1571 010224  032737  000200  177564          BIT     #200,@#DLVXSR   ;CHECK XMIT STATUS
1572 010232  001770                          BEQ     PUTCHR          ;NOT DONE YET
1573 010234  110237  177566                  MOVB    R2,@#DLVXBF     ;MOVE IT TO PRINT BUF
1574 010240  000207                          RTS     PC
1575
1576 010242  032737  000200  176504  CHCHR:  BIT     #200,@#CHXSR    ;CHECK XMIT STATUS
1577 010250  001774                          BEQ     CHCHR           ;NOT DONE YET
1578 010252  110237  176506                  MOVB    R2,@#CHXBF      ;MOVE IT TO PRINT BUF
1579 010256  000207                          RTS     PC
1580
1581                                                ;PRINT A DECIMAL INTEGER
1582 010260                          PRINT:
1583 010260  012703  000000G                 MOV     #STRING,R3      ;SETUP OUTPUT BUFFER
1584 010264  016502  000004                  MOV     4(R5),R2        ;GET HOW MANY DIGITS
1585 010270  017501  000002                  MOV     @2(R5),R1       ;GET NUMBER TO PRINT
1586 010274  005701                          TST     R1
1587 010276  002002                          BGE     5$              ;WORRY IF NEGATIVE
1588 010300  005401                          NEG     R1
1589 010302  005302                          DEC     R2              ;ONE LESS TO PRINT
1590 010304  006700                  5$:     SXT     R0              ;AND EXTEND TO 32 BITS
1591
1592 010306  071027  000012          10$:    DIV     #10.,R0         ;DIVIDE BY 10
1593 010312  062701  000060                  ADD     #60,R1          ;CONVERT REMAINDER TO ASCII
1594 010316  010123                          MOV     R1,(R3)+        ;AND STORE IN STRING
1595 010320  010001                          MOV     R0,R1           ;CONVERT QUOTIENT TO DIVIDEND
1596 010322  006700                          SXT     R0              ;AND TO 32 BITS
1597 010324  005302                          DEC     R2              ;CHECK NUMBER OF DIGITS
1598 010326  003367                          BGT     10$             ;GET ANOTHER
1599
1600 010330  005775  000002                  TST     @2(R5)          ;CHECK SIGN
1601 010334  002004                          BGE     20$
1602 010336  012702  000055                  MOV     #55,R2          ;NEGATIVE...
1603 010342  004767  177646                  JSR     PC,PUTCHR       ;...PRINT MINUS SIGN
1604
1605 010346  016501  000004          20$:    MOV     4(R5),R1        ;CHARACTERS TO PRINT
1606 010352  014302                  30$:    MOV     -(R3),R2        ;ADDRESS OF STRING (MSD)
1607 010354  004767  177634                  JSR     PC,PUTCHR       ;PRINT DIGIT
1608 010360  005301                          DEC     R1              ;DONE YET?
1609 010362  003373                          BGT     30$             ;NO
1610 010364  000207                          RTS     PC              ;YES
1611
1612
1613         000001                          .END
```

(2) MODULE A32D - Subroutine to control ADAC A/D converters

Calling Sequence:

CALL A32D (BUFFER, GAINS, NWORDS, ADS, GATED, CHNO, ISW)

where the arguments are:

BUFFER	Integer array to receive contiguous sampled data
GAINS	Integer array containing gains of individual A/D converters
NWORDS	Number of samples to digitize per channel
ADS	A/D setup word containing which A/D's in use; which to interrupt from
GATED	Integer variable specifying presence (1) or absence (0) of DRV11/toneburst gate
CHNO	Integer array containing port (channel) numbers for each channel
ISW	Integer variable for returned status

Internal Summary:

(LABEL)	(FUNCTION)	(PAGE)
SETUP ROUTINES		
A32D	Argument transfer; parameter checking	4
SETDMA	Load DMA bus address and word count registers; load pseudo-register for DMA control	4
SETA2D	Load pseudo-register for A/D control; load DMA pseudo-registers into control status registers	6
MEASUREMENT ROUTINE		
READY	Prepare for measurement	7
GO	Check for pulsed-system gate	7
GOCW	Load A/D pseudo registers into actual control status registers; wait for completion (interrupt)	7

ADISR	Dummy interrupt service routine	8
DMISR	Functional interrupt service routine	8

```
                        .TITLE  A32D
                        .IDENT  /07/
                        ;
                        ;SUBROUTINE TO LOAD AN INTEGER DATA ARRAY WITH DATA FROM UP
                        ;TO THREE ADAC-1012 A/D CONVERTERS, RUNNING SIMULTANEOUSLY.
                        ;
                        ;AUTHOR: RICHARD E. SCOTT, JR. (DECEMBER 1981)
                        ;
                        ;FORTRAN CALL--
                        ;CALL A32D (BUFFER,GAINS,NWORDS,ADS,CATED,CHNO,ISW)
                        ;
                        ;"BUFFER" IS AN INTEGER ARRAY TO CONTAIN CONTIGUOUS DATA FROM
                        ;UP TO THREE CHANNELS OF DMA'ED DATA FROM THE A/D CONVERTERS.
                        ;EACH DATA WORD IS A 12-BIT INTEGER. FOR N WORDS PER CHANNEL...
                        ;  CHANNEL 1 DATA IN BUFFER(1) TO BUFFER(N)
                        ;  CHANNEL 2 DATA IN BUFFER(N+1) TO BUFFER(2N)
                        ;  CHANNEL 3 DATA IN BUFFER(2N+1) TO BUFFER(3N)
                        ;
                        ;"GAINS" ARE PROGRAMMABLE GAIN CODES (1,2,4,8) OR (1,2,5,10)
                        ;FOR A/D CONVERTERS (AS DEFINED BY HARDWARE STRAPPING).
                        ;GAINS IS AN INTEGER ARRAY.
                        ;
                        ;"NWORDS" IS THE NUMBER OF WORDS PER CHANNEL TO BE DMA'ED
                        ;INTO MEMORY FROM THE A/D CONVERTERS.
                        ;
                        ;"ADS" IS THE A/D SETUP ARRAY WHERE:
                        ;  ADS(1) = A/D'S USED (#1=1) + (#2=2) + (#3=4)  [ALL=7]
                        ;  ADS(2) = DMA TO GET INTERRUPT FROM (1,2,3)
                        ;AGAIN, WITH ADS(1) = 3 ONE CAN USE TWO A/D CARDS ONLY WITHOUT
                        ;CONCERN FOR THE SUBROUTINE ADDRESSING THE NONEXISTENT THIRD
                        ;A/D CARDS.
                        ;
                        ;"CATED" FLAGS THE PRESENCE OF PULSE/GATING MECHANISMS.
                        ;CATED = 1 MEANS USE PARALLEL I/O CARD TO FLAG START OF PULSE.
                        ;CATED = 0 MEANS DON'T.
                        ;THE LATTER CASE MEANS THAT A PARTICULAR BACKPLANE DOES NOT
                        ;NEED TO HAVE A DRV-11 INSTALLED TO AVOID TRAPS DUE TO ADDRESSING
                        ;UNNEEDED AND NONEXISTENT CARDS.  HOWEVER, ONE MAY FIND OCCASIONAL
                        ;PHASE ERRORS BECAUSE OF CLOCK PULSES OCCURRING DURING THE SETUP
                        ;OF THE A/D REGISTERS.  ONE TECHNIQUE USED IS TO REQUIRE THAT THE
                        ;HOST PROCESSOR TURN OFF THE SAMPLING CLOCK UNTIL THE REGISTERS
                        ;ARE ALL PREPARED.
                        ;
                        ;"CHNO" IS AN ARRAY OF THREE CHANNEL NUMBERS, CHANNEL IN THE SENSE
                        ;OF THE SIXTEEN POSSIBLE MULTIPLEXED PORTS TO EACH A/D CONVERTER.
                        ;
                        ;"ISW" IS STATUS OF TRANSFER (0 = AOK)
                        ;ISW =   -1; A/D #1 ERROR (THESE 6 ERRORS ARE ADDITIVE)
                        ;ISW =   -2; A/D #2 ERROR (DITTO...)
                        ;ISW =   -4; A/D #3 ERROR
                        ;ISW =  -10; DMA #1 ERROR
                        ;ISW =  -20; DMA #2 ERROR
                        ;ISW =  -40; DMA #3 ERROR (...RANGING -1 TO -77)
                        ;ISW = -100; PARAMETER ERROR (FROM THIS PROGRAM: A/D SETUP ERROR)
                        ;ISW = -101; PARAMETER ERROR (FROM CALLING PROGRAM: TOO MANY POINTS)
                        ;
                        ;**** ALL ARGUMENTS ARE INTEGERS ****
```

```
                                        ;HARDWARE REGISTER DEFINITIONS


                                        ;REGISTERS FOR THE FIRST A/D CONVERTER (A):
164040                                  ADCSRA=164040   ;A/D CONTROL STATUS REGISTER
164042                                  ADDBRA=164042   ;A/D DATA BUFFER
172410                                  DMWCRA=172410   ;DMA WORD COUNT REGISTER
172412                                  DMCARA=172412   ;DMA BUS ADDRESS REGISTER
172414                                  DMCSRA=172414   ;DMA STATUS REGISTER
172416                                  DMMUXA=172416   ;DMA CHANNEL DATA

                                        ;REGISTERS FOR THE SECOND A/D CONVERTER (B):
164050                                  ADCSRB=164050   ;A/D CONTROL STATUS REGISTER
164052                                  ADDBRB=164052   ;A/D DATA BUFFER
172420                                  DMWCRB=172420   ;DMA WORD COUNT REGISTER
172422                                  DMCARB=172422   ;DMA BUS ADDRESS REGISTER
172424                                  DMCSRB=172424   ;DMA STATUS REGISTER
172426                                  DMMUXB=172426   ;DMA CHANNEL DATA

                                        ;REGISTERS FOR THE THIRD A/D CONVERTER (C):
164060                                  ADCSRC=164060   ;A/D CONTROL STATUS REGISTER
164062                                  ADDBRC=164062   ;A/D DATA BUFFER
172430                                  DMWCRC=172430   ;DMA WORD COUNT REGISTER
172432                                  DMCARC=172432   ;DMA BUS ADDRESS REGISTER
172434                                  DMCSRC=172434   ;DMA STATUS REGISTER
172436                                  DMMUXC=172436   ;DMA CHANNEL DATA

                                        ;A/D CONTROL STATUS REGISTER (ADCSR*):
                                        ;BIT 15:    ERROR BIT (SET FOR CONVERSION TIMING ERROR)
                                        ;BIT 14:    INTERRUPT ON ERROR (0)
                                        ;BIT 12-3: NOT USED (EXTENDED CHANNEL ADDRESS)
                                        ;BIT 11-8: ADDRESS OF CHANNEL (*DD)
                                        ;BIT  7:    A/D DONE FLAG (SET AT END OF CONVERSION)
                                        ;BIT  6:    INTERRUPT ENABLE (0)
                                        ;BIT  5:    OPTIONAL ENABLE (0)
                                        ;BIT  4-3: PROGRAMMABLE GAIN (AS INPUT)
                                        ;BIT  2:    SEQUENTIAL ENABLE
                                        ;BIT  1:    EXTERNAL ENABLE (1)
                                        ;BIT  0:    A/D CONVERSION GO BIT (0)

                                        ;DATA REGISTER (ADDBR*):
                                        ;BITS 12-15: NOT USED (1111)
                                        ;BITS 00-11: DATA

                                        ;DMA WORD COUNT REGISTER (DMWCR*)
                                        ;BITS 12-15: NOT USED (1111)
                                        ;BITS 00-11: ONE'S COMPLEMENT OF WORDS

                                        ;DMA BUS ADDRESS (DMCAR*):
                                        ;BITS 00-11: WORD ADDRESS
```

```
                                ;DMA CONTROL STATUS REGISTER (DMCSR*):
                                ;BIT 15:    ERROR BIT (SET FOR ANY ERROR)
                                ;BIT 14:    NONEXISTENT MEMORY ERROR
                                ;BIT 13:    ATTENTION BIT (EXTERNAL)
                                ;BIT 12-8:  NOT USED (0)
                                ;BIT  7:    READY FLAG (SET WHEN DMA DONE)
                                ;BIT  6:    INTERRUPT ENABLE (0)
                                ;BIT  5-4:  EXTENDED ADDRESS BITS
                                ;BIT  3-1:  NOT USED (0)
                                ;BIT  0:    DMA GO BIT (1)


                                ;REGISTERS FOR THE DRV-11 USED FOR SYNCHRONIZATION IN PULSED
                                ;SYSTEMS.
                                ;
167770                          DRCSR=167770
167772                          DROUT=167772
167774                          DRINN=167774

                                ;VECTOR ADDRESSES:
                                ;    130 - A/D COMPLETION (A) OR ERROR
                                ;    150 - DMA COMPLETION (A)
                                ;    170 - A/D COMPLETION (B) OR ERROR
                                ;    210 - DMA COMPLETION (B)
                                ;    230 - A/D COMPLETION (C) OR ERROR
                                ;    250 - DMA COMPLETION (C)

                                ;SET PROM = 1 IF VECTOR SPACE IS WITHIN ROM SPACE AND SET
                                ;VECTORS REQUIRED. OTHERWISE, VECTORS ARE LOADED DYNAMICALLY.
                                ;SET PROM = -1 FOR RAM VARIABLES ASSEMBLED LOCALLY.
000001                          PROM=1
```

```
                                        ;SUBROUTINE INITIALIZATION:
                                        ;  GRAB THE ARGUMENTS.
                                        ;  DO SOME CHECKING FOR VALIDITY.

000000                          A32D::                          ;START OF ROUTINE

000000  005775  000012                  TST     @12(R5)         ;IF HAVE, THEN...
000004  001402                          BEQ     10$
000006  005037  167770                  CLR     @#DRCSR         ;CLEAR UP THE DRV-11 STATUS REGISTER

000012  016500  000010          10$:    MOV     10(R5),R0       ;ADDRESS OF ADS
000016  011067  000000G                 MOV     (R0),WAD        ;A/D'S-USED WORD
000022  016067  000002  000000G         MOV     2(R0),IWD       ;DMA INTERRUPT WORD
000030  016500  000002                  MOV     2(R5),R0        ;ADDRESS OF FIRST DATA BUFFER
000034  017504  000006                  MOV     @6(R5),R4       ;NUMBER OF WORDS
000040  005104                          COM     R4              ;BUT ONE'S COMPLEMENTED
000042  016502  000014                  MOV     14(R5),R2       ;ADDRESS OF CHANNEL PORTS

000046  022767  000007  000000G         CMP     #7,WAD          ;CHECK FOR ERRORS
000054  002414                          BLT     PERR            ;WAD > 7
000056  022767  000001  000000G         CMP     #1,WAD
000064  003010                          BGT     PERR            ;WAD < 1
000066  022767  000003  000000G         CMP     #3,IWD
000074  002404                          BLT     PERR            ;IWD > 3
000076  022767  000001  000000G         CMP     #1,IWD
000104  003404                          BLE     SETDMA          ;IWD >=1

000106  012775  177700  000016  PERR:   MOV     #-100,@16(R5)   ;PARAMETER ERROR
000114  000207                          RTS     PC

                                        ;SET UP THE NECESSARY REGISTERS FOR THOSE A/D-DMAS
                                        ;THAT ARE TO BE USED.

                                        ;FIRST DMA REGISTERS SETUP:

000116  032767  000001  000000G SETDMA: BIT     #1,WAD          ;IS THIS DMA ACTIVE?
000124  001424                          BEQ     20$             ;NOPE!
000126  010437  172410                  MOV     R4,@#DMWCRA     ;LOAD WORD COUNT REGISTER
000132  010037  172412                  MOV     R0,@#DMCARA     ;LOAD ADDRESS REGISTER
000136  012737  000001  000000G         MOV     #1,@#TI         ;LOAD CSR, GO ONLY
000144  012201                          MOV     (R2)+,R1        ;GET PORT NUMBER
000146  010137  172416                  MOV     R1,@#DMMUXA     ;LOAD DMA MULTIPLEXOR ADDRESS
000152  000301                          SWAB    R1              ;SHIFT INTO BITS 8-11
000154  010137  060000                  MOV     R1,@#ADD        ;THEN STORE FOR LATER USE
                                        .IF EQ PROM
                                        MOV     #ADISR,@#130    ;LOAD INTERRUPT VECTOR (A/D)
                                        MOV     #340,@#132      ;AND PROCESSOR STATUS WORD
                                        MOV     #DMISR,@#150    ;LOAD INTERRUPT VECTOR (DMA)
                                        MOV     #340,@#152      ;AND PROCESSOR STATUS WORD
                                        .ENDC
000160  022767  000001  000000G         CMP     #1,IWD          ;INTERRUPT OFF THIS ONE?
000166  001003                          BNE     20$             ;NOPE!
000170  012737  000101  000000G         MOV     #101,@#TI       ;LOAD CSR, IE AND GO
                                        .IF EQ PROM
                                        MOV     #DMISR,@#150    ;LOAD ACTIVE ISR VECTOR
                                        .ENDC
```

```
                                                ;SECOND DMA REGISTERS SETUP
                                                
000176  067500  000006          20$:  ADD     @6(R5),R0       ;INCREASE BUFFER ADDRESS
000202  067500  000006                ADD     @6(R5),R0       ;(TWICE FOR BYTES)
000206  032767  000002  000000G       BIT     #2,WAD          ;IS THIS DMA ACTIVE?
000214  001424                        BEQ     30$             ;NOPE!
000216  010437  172420                MOV     R4,@#DMWCRB     ;LOAD WORD COUNT REGISTER
000222  010037  172422                MOV     R0,@#DMCARB     ;LOAD ADDRESS REGISTER
000226  012737  000001  000000G       MOV     #1,@#T2         ;LOAD CSR, GO ONLY
000234  012201                        MOV     (R2)+,R1        ;GET PORT NUMBER
000236  010137  172426                MOV     R1,@#DMMUXB     ;LOAD DMA MULTIPLEXOR ADDRESS
000242  000301                        SWAB    R1              ;SHIFT INTO BITS 8-11
000244  010137  000000G               MOV     R1,@#BDD        ;THEN STORE FOR LATER USE
                                      .IF EQ PROM
                                      MOV     #ADISR,@#170    ;LOAD INTERRUPT VECTOR (A/D)
                                      MOV     #340,@#172      ;AND PROCESSOR STATUS WORD
                                      MOV     #DMISR,@#210    ;LOAD INTERRUPT VECTOR
                                      MOV     #340,@#212      ;AND PROCESSOR STATUS WORD
                                      .ENDC
000250  022767  000002  000000G       CMP     #2,IWD          ;INTERRUPT OFF THIS ONE?
000256  001003                        BNE     30$             ;NOPE!
000260  012737  000101  000000G       MOV     #101,@#T2       ;LOAD CSR, IE AND GO
                                      .IF EQ PROM
                                      MOV     #DMISR,@#210    ;LOAD ACTIVE ISR VECTOR
                                      .ENDC

                                                ;THIRD DMA REGISTERS SETUP

000266  067500  000006          30$:  ADD     @6(R5),R0       ;INCREASE BUFFER ADDRESS
000272  067500  000006                ADD     @6(R5),R0       ;(TWICE FOR BYTES)
000276  032767  000004  000000G       BIT     #4,WAD          ;IS THIS DMA ACTIVE?
000304  001426                        BEQ     SETA2D          ;NOPE!
000306  010437  172430                MOV     R4,@#DMWCRC     ;LOAD WORD COUNT REGISTER
000312  010037  172432                MOV     R0,@#DMCARC     ;LOAD ADDRESS REGISTER
000316  012737  000001  000000G       MOV     #1,@#T3         ;LOAD CSR, GO ONLY
000324  012201                        MOV     (R2)+,R1        ;GET PORT NUMBER
000326  010137  172436                MOV     R1,@#DMMUXC     ;LOAD DMA MULTIPLEXOR ADDRESS
000332  000301                        SWAB    R1              ;SHIFT INTO BITS 8-11
000334  042701  170377                BIC     #170377,R1      ;CLEAN UP ACCIDENTS
000340  010137  000000G               MOV     R1,@#CDD        ;THEN STORE FOR LATER USE
                                      .IF EQ PROM
                                      MOV     #ADISR,@#230    ;LOAD INTERRUPT VECTOR (A/D)
                                      MOV     #340,@#232      ;AND PROCESSOR STATUS WORD
                                      MOV     #DMISR,@#250    ;LOAD INTERRUPT VECTOR
                                      MOV     #340,@#252      ;AND PROCESSOR STATUS WORD
                                      .ENDC
000344  022767  000003  000000G       CMP     #3,IWD          ;INTERRUPT OFF THIS ONE?
000352  001003                        BNE     SETA2D          ;NOPE!
000354  012737  000101  000000G       MOV     #101,@#T3       ;LOAD CSR, IE AND GO
                                      .IF EQ PROM
                                      MOV     #DMISR,@#250    ;LOAD ACTIVE ISR VECTOR
                                      .ENDC
```

```
                                        ;PREPARE AND LOAD THE A/D CSR'S WITH GAIN, ETC.--EVERYTHING
                                        ;EXCEPT THE GO BITS. ALSO LOAD THE DMA CSR'S PREVIOUSLY CHOSEN.
                                        ;LOAD R0-R3 WITH THE ADDRESSES OF THE ACTIVE A/D CSR'S (OR A
                                        ;DUMMY ADDRESS IF NOT).  THIS WILL ALLOW QUICK SETUP-AND-GO
                                        ;OF THE A/D'S (NEAR SIMULTANEOUS) WITHOUT THE LOGIC REQUIRED
                                        ;TO NOT "DO" A/D CSR'S NOT EXISTING.

000362  016503  000004          SETA2D: MOV     4(R5),R3        ;ADDRESS OF GAINS ARRAY
000366  017567  000010  000000G        MOV     @10(R5),WAD     ;GET A/D APPLICABILITY WORD

                                        ;SET UP A/D CONVERTER NUMBER ONE IF ACTIVE

000374  012700  000000G          10$:  MOV     #ADUM,R0        ;TARGET ADDRESS IF NOT ACTIVE
000400  032767  000001  000000G        BIT     #1,WAD          ;IS THIS A/D ACTIVE?
000406  001425                        BEQ     30$             ;NOPE
000410  012300                        MOV     (R3)+,R0        ;GET GAIN CODE OF CHANNEL A (1,2,5,10)
000412  006200                        ASR     R0              ;CHANGE TO 0,1,2,5
000414  022700  000005                CMP     #5,R0           ;CHECK IF 5
000420  003002                        BGT     20$             ;NOPE
000422  005300                        DEC     R0              ;SET 5 TO 4
000424  005300                        DEC     R0              ;THEN TO 3
000426  072027  000003          20$:  ASH     #3,R0           ;SHIFTED INTO BITS 3-4
000432  005100                        COM     R0              ;SET TO 30,20,10,00
000434  042700  177747                BIC     #177747,R0      ;SAFETY CLEANUP
000440  053700  060000                BIS     @#ADD,R0        ;ADD ADDRESS SETTING TO R0
000444  010037  164040                MOV     R0,@#ADCSRA     ;SET GAIN (ETC) BITS FIRST
000450  012700  164040                MOV     #ADCSRA,R0      ;TARGET ADDRESS IF ACTIVE
000454  016737  000000G 172414        MOV     T1,@#DMCSRA     ;ENABLE DMA'S

                                        ;SET UP A/D CONVERTER NUMBER TWO IF ACTIVE

000462  012701  000000G          30$:  MOV     #ADUM,R1        ;TARGET ADDRESS IF NOT ACTIVE
000466  032767  000002  000000G        BIT     #2,WAD          ;IS THIS A/D ACTIVE?
000474  001425                        BEQ     50$             ;NOPE
000476  012301                        MOV     (R3)+,R1        ;GET GAIN CODE OF CHANNEL B (1,2,5,10)
000500  006201                        ASR     R1              ;CHANGE TO 0,1,2,5
000502  022701  000005                CMP     #5,R1           ;CHECK IF 5
000506  003002                        BGT     40$             ;NOPE
000510  005301                        DEC     R1              ;SET 5 TO 3
000512  005301                        DEC     R1
000514  072127  000003          40$:  ASH     #3,R1           ;SHIFTED INTO BITS 3-4
000520  005101                        COM     R1              ;SET TO 30,20,10,00
000522  042701  177747                BIC     #177747,R1
000526  053701  000000G                BIS     @#BDD,R1        ;ADD ADDRESS SETTING TO R1
000532  010137  164050                MOV     R1,@#ADCSRB     ;SET GAIN (ETC) BITS FIRST
000536  012701  164050                MOV     #ADCSRB,R1      ;TARGET ADDRESS IF ACTIVE
000542  016737  000000G 172424        MOV     T2,@#DMCSRB     ;ENABLE DMA'S
```

```
                                        ;SET UP A/D CONVERTER NUMBER THREE IF ACTIVE
                                        ;
000550  012702  000000G         50$:    MOV     #ADUM,R2        ;TARGET ADDRESS IF NOT ACTIVE
000554  032767  000004  000000G         BIT     #4,WAD          ;IS THIS A/D ACTIVE?
000562  001425                         BEQ     READY           ;NOPE
000564  012302                         MOV     (R3)+,R2        ;GET GAIN CODE OF CHANNEL C (1,2,5,10)
000566  006202                         ASR     R2              ;CHANGE TO 0,1,2,5
000570  022702  000005                 CMP     #5,R2           ;CHECK IF 5
000574  003002                         BGT     60$             ;NOPE
000576  005302                         DEC     R2              ;SET 5 TO 3
000600  005302                         DEC     R2
000602  072227  000003         60$:    ASH     #3,R2           ;SHIFTED INTO BITS 3-4
000606  005102                         COM     R2              ;SET TO 30,20,10,00
000610  042702  177747                 BIC     #177747,R2
000614  053702  000000G                BIS     @#CDD,R2        ;ADD ADDRESS SETTING TO R2
000620  010237  164060                 MOV     R2,@#ADCSRC     ;SET GAIN (ETC) BITS FIRST
000624  012702  164060                 MOV     #ADCSRC,R2      ;TARGET ADDRESS IF ACTIVE
000630  016737  000000G 172434         MOV     T3,@#DMCSRC     ;ENABLE DMA'S
                                        ;
                                        ;
                                        ;IS EVERYBODY READY???
                                        ;INCLUDING TRANSMIT GATE TO DLV-11???
                                        ;
000636  005067  000000G         READY:  CLR     ADST            ;CLEAR STATUS WORD
000642  005075  000016                 CLR     @16(R5)         ;CLEAR ERROR STATUS
                                        ;
000646  000240                         NOP
000650  005775  000012                 TST     @12(R5)         ;USE DRV-11 FOR PULSE GATE?
000654  001404                         BEQ     COCW            ;NO
000656  032737  000001  167774 CO:     BIT     #1,@#DRINN      ;CHECK FOR BIT 0 SET
000664  001774                         BEQ     CO              ;...AND WAIT...
                                        ;
                                        ;THE FOLLOWING IS DONE THIS WAY SO THE LOADING IS AS FAST
                                        ;AS POSSIBLE, THUS ELIMINATING ANY POSSIBLE PHASE ERRORS
                                        ;BETWEEN THE THREE CHANNELS.
                                        ;TIME TO LOAD EACH CSR IS ABOUT 4.54 MICROSECONDS.
                                        ;NOTE THAT IF ACTIVE, THE DESTINATION REGISTERS CONTAIN THE
                                        ;ADDRESSES OF THE A/D CSR'S; IF NOT ACTIVE, THEY CONTAIN A
                                        ;DUMMY ADDRESS TO AVOID THE POSSIBILITY OF WRITING TO A
                                        ;NON-EXISTENT ADDRESS (AND TRAPPING) IF INACTIVE A/D CONVERTERS
                                        ;ARE NOT IN THE BACKPLANE.
                                        ;
000666  012703  000003         COCW:   MOV     #3,R3
000672  050310                         BIS     R3,(R0)         ;START A/D CONVERSIONS
000674  050311                         BIS     R3,(R1)
000676  050312                         BIS     R3,(R2)
                                        ;
000700  000001                 20$:    WAIT
000702  005767  000000G                TST     ADST            ;WAIT FOR SPECIFIED INTERRUPT
000706  000240                         NOP
000710  001773                         BEQ     20$             ;...AND WAIT...
                                        ;
000712  000207                         RTS     PC              ;RETURN WITH DATA
```

```
                                        ;INTERRUPT SERVICE ROUTINE
                                        ;   DMISR -- DMA SERVICE ROUTINE
                                        ;   ADISR -- A/D SERVICE ROUTINE
                                        ;WITH SIMULTANEOUS ACTION, ALL SHOULD FINISH AT ONE TIME, THUS
                                        ;NO NEED TO GO THROUGH ISR FOR ALL THREE DEVICES.
                                        ;(INTERRUPTS DISABLED ON REMAINING DEVICES ANYWAY.)

000714  000002              ADISR:: RTI             ;;;DO NOTHING

000716  012767  000001  000000G DMISR:: MOV     #1,ADST ;;;GOOD INTERRUPT: EOR FOR CHANNEL SPECIFIED

000724  032767  000001  000000G         BIT     #1,WAD          ;;;THIS A/D ACTIVE?
000732  001414                         BEQ     20$             ;;;NOPE
000734  005737  164040                 TST     @#ADCSRA        ;;;ERRORS FOR CHANNEL A?
000740  100003                         BPL     10$             ;;;NO
000742  062775  177777  000016         ADD     #-1,@16(R5)     ;;;YES: ERROR -1+
000750  005737  172414           10$:  TST     @#DMCSRA        ;;;ERRORS FOR CHANNEL A?
000754  100003                         BPL     20$             ;;;NO
000756  062775  177770  000016         ADD     #-10,@16(R5)    ;;;YES: ERROR -10+
000764  032767  000002  000000G 20$:   BIT     #2,WAD          ;;;IS THIS A/D ACTIVE?
000772  001414                         BEQ     40$             ;;;NOPE!
000774  005737  164050                 TST     @#ADCSRB        ;;;ERRORS FOR CHANNEL B?
001000  100003                         BPL     30$             ;;;NO
001002  062775  177776  000016         ADD     #-2,@16(R5)     ;;;YES: ERROR -2+
001010  005737  172424           30$:  TST     @#DMCSRB        ;;;ERRORS FOR CHANNEL B?
001014  100003                         BPL     40$             ;;;NO
001016  062775  177760  000016         ADD     #-20,@16(R5)    ;;;YES: ERROR -20+
001024  032767  000004  000000G 40$:   BIT     #4,WAD          ;;;IS THIS A/D ACTIVE?
001032  001414                         BEQ     60$             ;;;NOPE!
001034  005737  164060                 TST     @#ADCSRC        ;;;ERRORS FOR CHANNEL C?
001040  100003                         BPL     50$             ;;;NO
001042  062775  177774  000016         ADD     #-4,@16(R5)     ;;;YES: ERROR -4+
001050  005737  172434           50$:  TST     @#DMCSRC        ;;;ERRORS FOR CHANNEL C?
001054  100003                         BPL     60$             ;;;NO
001056  062775  177740  000016         ADD     #-40,@16(R5)    ;;;YES: ERROR -40+
001064  000002              60$:  RTI

                                        .IF LT PROM
                                        ;RAM VARIABLES USED BY A32D
                                WAD:    .WORD   0
                                IWD:    .WORD   0
                                T1:     .WORD   0
                                T2:     .WORD   0
                                T3:     .WORD   0
                                ADUM:   .WORD   0
                                ADST:   .WORD   0
                                ADD:    .WORD   0
                                BDD:    .WORD   0
                                CDD:    .WORD   0
                                        .ENDC

        000001                          .END
```

(3) MODULE CONVRT - Subroutine to convert a floating-point number to an ASCII string.

Calling Sequence:

```
CALL CONVRT     (VALUE, STRING, K, L)
```

where the arguments are:

VALUE	Floating-point variable to be encoded
STRING	Byte array to load encoded variable
K	Encode with K places to left of decimal point
L	Encode with L places to right of decimal point

NOTE: Decimal point (.) and sign (+ or -) are not included in the size of K and L.

```
                                        .TITLE  CONVRT
                                        .IDENT  /01/

                                        ;SUBROUTINE TO CONVERT A REAL NUMBER (FLOATING) TO
                                        ;AN ASCII STRING OF FORMAT "KK.LLL" FOR PRINTING
                                        ;PURPOSES.
                                        ;
                                        ;CALL CONVRT (VALUE,STRING,K,L)
                                        ;WHERE VALUE BECOMES AS F(K).(L) (FORTRAN)
                                        ;
                                        ;AUTHOR: RICHARD E. SCOTT, JR. (NRL/USRD -- AUGUST 1981)

                000000                  AC0=%0
                000001                  AC1=%1

000000                          CONVRT::
000000  010046                          MOV     R0,-(SP)        ;SAVE REGISTERS ON STACK
000002  010146                          MOV     R1,-(SP)
000004  010246                          MOV     R2,-(SP)
000006  010346                          MOV     R3,-(SP)
000010  010446                          MOV     R4,-(SP)

000012  170001                          SETF                    ;SET TO FLOATING MODE
000014  016503  000004                  MOV     4(R5),R3        ;ADDRESS OF OUTPUT STRING
000020  172475  000002                  LDF     @2(R5),AC0      ;GET VALUE TO PRINT
000024  170000                          CFCC                    ;POSITIVE OR NEGATIVE?
000026  003003                          BGT     10$
000030  112723  000055                  MOVB    #'-,(R3)+       ;INSERT MINUS SIGN
000034  000402                          BR      20$
000036  112723  000053          10$:    MOVB    #'+,(R3)+       ;INSERT PLUS SIGN
000042  170600                  20$:    ABSF    AC0             ;ABSOLUTE VALUE
000044  175401                          STCFI   AC0,R1          ;SAVE INTEGER PART (VOLTS?)
000046  177101                          LDCIF   R1,AC1
000050  017504  000006                  MOV     @6(R5),R4       ;DIGITS LEFT OF DECIMAL
000054  004767  000054                  JSR     PC,DECO         ;LOAD THEM IN STRING
000060  112723  000056                  MOVB    #'.,(R3)+       ;AND DECIMAL POINT
000064  173001                          SUBF    AC1,AC0         ;FIND FRACTION PART (MILLIVOLTS)
000066  017504  000010                  MOV     @10(R5),R4
000072  171037  000166'          30$:    MULF    @#TEN,AC0       ;CONVERT TO 10**L INTEGER
000076  005304                          DEC     R4
000100  003374                          BGT     30$
000102  172037  000172'                  ADDF    @#RO,AC0        ;FIX ROUND-OFF
000106  175401                          STCFI   AC0,R1          ;SAVE AS INTEGER
000110  017504  000010                  MOV     @10(R5),R4      ;DIGITS RIGHT OF DECIMAL
000114  004767  000014                  JSR     PC,DECO         ;LOAD THEM IN STRING

000120  012604                          MOV     (SP)+,R4        ;RESTORE REGISTERS ON STACK
000122  012603                          MOV     (SP)+,R3
000124  012602                          MOV     (SP)+,R2
000126  012601                          MOV     (SP)+,R1
000130  012600                          MOV     (SP)+,R0

000132  000207                          RTS     PC
```

```
55 000134                              DECO:                           ;DATA IN R1
56 000134  010402                              MOV     R4,R2           ;CHARACTER COUNTER
57 000136  060203                              ADD     R2,R3           ;POINTER--WORK RIGHT TO LEFT
58 000140  006700                      10$:    SXT     R0              ;EXTEND VALUE TO 32 BITS
59 000142  071027  000012                       DIV     #10.,R0         ;DIVIDE BY 10
60 000146  062701  000060                       ADD     #60,R1          ;CONVERT REMAINDER TO ASCII
61 000152  110143                               MOVB    R1,-(R3)        ;AND STORE IN STRING
62 000154  010001                               MOV     R0,R1           ;CONVERT QUOTIENT TO DIVIDEND
63 000156  005302                               DEC     R2              ;DONE?
64 000160  003367                               BGT     10$             ;NO
65 000162  060403                               ADD     R4,R3           ;RESET STRING POINTER
66 000164  000207                               RTS     PC
67
68 000166  041040  000000              TEN:    .FLT2   10.
69 000172  037777  171345              R0:     .FLT2   0.4999
70
71         000001                               .END
```

(4) MODULE DAPD - Subroutine to compute voltage and phase from the real and imaginary parts of the voltage, and to compute the harmonic distortion from a specific number of harmonics.

Calling Sequence:

CALL DAPD (X, N, K, AMP, PHASE, DSTORT, LHARM, IERR)

where the arguments are:

X	Floating-point data sequence
N	Length of data sequence
K	Spectral line (number of cycles) to compute
AMP	Resulting total rms voltage
PHASE	Resulting phase in degrees
DSTORT	Harmonic distortion
LHARM	Number of harmonics used to compute distortion
IERR	Execution status

NOTES:
1. If LHARM 1, distortion is not computed
2. If DSTORT = -1, distortion was not computed
3. If DSTORT = -2, error occurred (AMP = 0)
4. If IERR = -1, K = N (illegal condition)
5. If IERR = LHARM, harmonic folds onto dc line

```
                                        .TITLE  DAPD
                                        .IDENT  /01/

                                        ;AMPPHA.FTN 17-JUL-81 JDG GEORGE (ORIGINATOR)
                                        ;DAPD  .MAC 02-OCT-81 RE SCOTT (TRANSLATOR)

                                        ;SUBROUTINE DAPD (X,N,K,AMP,PHASE,DSTORT,LHARM,IERR)
                                        ;
                                        ;COMPUTE AMPLITUDE, PHASE, & PERCENT HARMONIC DISTORTION
                                        ;FOR THE K-TH SPECTRAL LINE OF THE DFT OF X, A REAL SEQUENCE
                                        ;OF LENGTH N.
                                        ;UNDERSAMPLING OR FOLDING WHERE K CAN EXCEED N/2 IS ALLOWED.
                                        ;SUBROUTINES REQUIRED:  DGZL (X,N,K,XR,XI)

                                        ;REAL X(N)       !DATA SEQUENCE
                                        ;INTEGER N       !SEQUENCE LENGTH
                                        ;INTEGER K       !DFT SPECTRAL LINE NUMBER OF INTEREST
                                        ;                !ALSO # CYCLES OF TEST SINUSOID IN N SAMPLES
                                        ;                !I.E. K CYCLES OF FUNDAMENTAL (1ST HARM) IN SEQUENCE;
                                        ;                !  2K CYCLES OF 2ND HARMONIC, ETC.
                                        ;                !IF K=0, COMPUTE DC COMPONENT.
                                        ;REAL AMP        !RMS MAGNITUDE OF K TH SPECTRAL COMPONENT
                                        ;REAL PHASE      !PHASE IN DEGREES OF K-TH SPECTRAL COMPONENT
                                        ;                !MODEL IS 0 DEG COSINE -90 DEG SINE
                                        ;REAL DSTORT     !% HARMONIC DISTORTION OF K TH COMPONENT
                                        ;                !FOR LINE #'S 2*K THROUGH LHARM*K
                                        ;INTEGER LHARM   !MAX # HARMONIC TO INCLUDE IN DISTORTION
                                        ;                !IF LHARM=0 SKIP DISTORTION CALCULATION; SAVE TIME
                                        ;INTEGER IERR    !IERR=0 NORMAL
                                        ;                !IERR=-1 IF(MOD(K,N)=0
                                        ;                !IERR=-L IF HARMONIC FOLDS ON DC LINE

                                        ;DSTORT = -1 IF NOT COMPUTED
                                        ;DSTORT = -2 IF ERROR IN COMPUTING

000000                                  AC0=%0
000001                                  AC1=%1
000002                                  AC2=%2

000013                                  HARLIN = 11.     ;MAXIMUM HARMONICS TO USE

                                        ;EXPERIENCE SEEMS TO SHOW THAT A MINIMUM OF 9 POINTS ARE
                                        ;REQUIRED TO COMPUTE ACCURATELY THE AMPLITUDE AND PHASE OF
                                        ;A SINUSOID; FOR DISTORTION, 2*LHARM+1 POINTS ARE REQUIRED.
                                        ;
                                        ;SET PROM = 1 FOR RAM VARIABLES STORED EXTERNALLY
                                        ;SET PROM = 0 FOR RAM VARIABLES STORED INTERNALLY (NORMAL)
000001                                  PROM=1

000000                          DAPD::
000000  010046                        MOV     R0,-(SP)                ;SAVE REGISTERS ON STACK
000002  010146                        MOV     R1,-(SP)
000004  010246                        MOV     R2,-(SP)
000006  010346                        MOV     R3,-(SP)
000010  010446                        MOV     R4,-(SP)

000012  170C91                        SETF
```

```
                                        ;*****************************************
                                        ;        COMPUTE AMPLITUDE AND PHASE
                                        ;*****************************************

                                                                ;*;IERR=0
000014  005075  000020                  CLR     @20(R5)
                                                                ;*;I=MOD(K,N)
000020  017501  000006                  MOV     @6(R5),R1               ;NCYC = CYCLES PER SAMPLE
000024  006700                          SXT     R0
000026  071075  000004                  DIV     @4(R5),R0               ;NCYC / NSAM
                                                                ;*;IF((I.EQ.0).AND.(K.NE.0)) IERR=-1
000032  005701                          TST     R1                      ;R1 = I (REMAINDER)
000034  001006                          BNE     15$
000036  005775  000006                  TST     @6(R5)                  ;K
000042  001403                          BEQ     15$
000044  012775  177777  000020          MOV     #-1,@20(R5)             ;IERR = -1
000052  017502  000004          15$:    MOV     @4(R5),R2               ;NSAM = SAMPLES PER SEQUENCE
000056  010237  000000G                 MOV     R2,@#NSAM               ;SAVE FOR DGZL
000062  006202                          ASR     R2                      ;NSAM / 2
                                                                ;*;IF (I.LE.N/2) I=I
                                                                ;*;IF (I.LE.N/2) PSIGN=+1
000064  020102                          CMP     R1,R2                   ;COMPARE I AND NSAM/2
000066  003006                          BGT     20$                     ;I > NSAM/2
000070  010137  000000G                 MOV     R1,@#I                  ;I =< NSAM/2 SO KEEP I
000074  012737  000001  000000G         MOV     #1,@#PSIGN              ;NO PHASE SIGN-CHANGE
000102  000416                          BR      30$
                                                                ;*;IF (I.GT.N/2) I=N-I
                                                                ;*;IF (I.GT.N/2) PSIGN=-1
000104  167501  000004          20$:    SUB     @4(R5),R1               ;GET I - NSAM
000110  005401                          NEG     R1                      ;NSAM - I
000112  010137  000000G                 MOV     R1,@#I                  ;USE I = NSAM-I
000116  012737  177777  000000G         MOV     #-1,@#PSIGN             ;PREPARE TO CHANGE-SIGN OF PHASE
                                                                        ;(WAGON WHEEL REVERSAL MODE)

                                                                ;*;HARM(1)=I
000124  012700  000000G         30$:    MOV     #HARM,R0                ;GET ADDRESS OF HARMONIC ARRAY
000130  010110                          MOV     R1,(R0)                 ;HARM(1) = I (THIS ONE DONE)

                                                                ;*;CALL DGRTZL (X,N,K,XR,XI)
                                                                        ;PROCESS FUNDAMENTAL
000132  010546                          MOV     R5,-(SP)                ;SAVE R5 (DURING CALL DGZL)
                                        .IF EQ PROM
                                        MOV     2(R5),@#DGARG+2         ;TRANSFER ADDRESS OF ARRAY
                                        .ENDC
000134  012705  000654'                 MOV     #DGARG,R5
000140  004767  000000G                 JSR     PC,DGZL                 ;DO SINGLE-POINT DFT
```

```
                                                                ;*;AMP=SQRT(2*(XR*XR+XI*XI))
000144  170001                CVAP:   SETF                    ;COMPUTE VOLTAGE AND PHASE
000146  172475  000010                LDF     @10(R5),AC0     ;XR
000152  174002                        STF     AC0,AC2         ;SAVE FOR TEST
000154  171000                        MULF    AC0,AC0         ;XR*XR
000156  172575  000012                LDF     @12(R5),AC1     ;XI
000162  171101                        MULF    AC1,AC1         ;XI*XI
000164  172001                        ADDF    AC1,AC0         ;XR*XR + XI*XI
000166  172000                        ADDF    AC0,AC0         ;2 (XR*XR + XI*XI)
000170  004767  000000G               JSR     PC,$$SQRT       ;=VDFT
000174  012605                        MOV     (SP)+,R5        ;RESTORE R5
000176  174075  000010                STF     AC0,@10(R5)
                                                                ;WATCH FOR ATAN(XI/0)
000202  170502                        TSTF    AC2
000204  170000                        CFCC
000206  001420                        BEQ     15$             ;ERROR!
                                                                ;*;IF(XR.NE.0)
                                                                ;*;PHASE=PSIGN*ATAN2(XI,XR)*180./3.1415927
000210  010546                        MOV     R5,-(SP)        ;SAVE R5 (FOR CALL SATAN2)
000212  012705  000720'              MOV     #ARGBLK,R5
000216  004767  000000G               JSR     PC,$ATAN2       ;TAN-1 (XI/XR)
000222  172400                        LDF     R0,AC0
000224  171037  000670'              MULF    @#ANGC,AC0      ;CONVERT TO DEGREES
000230  005767  000000G               TST     PSIGN           ;WAGON WHEELS?
000234  003001                        BGT     10$             ;FORWARD
000236  170700                        NEGF    AC0             ;NO--BACKWARD ROTATING
000240  012605                10$:    MOV     (SP)+,R5        ;RESTORE R5
000242  174075  000012                STF     AC0,@12(R5)     ;PDFT
000246  000404                        BR      CD1S
                                                                ;*;IF(XR.EQ.0) PHASE=0.0
000250  172637  000700'        15$:    LDF     @#P90,AC2       ;FOR ATAN(XI/0)...
000254  174275  000012                STF     AC2,@12(R5)     ;SET PDFT = 90
```

```
                                        ;*****************************************
                                        ;       COMPUTE HARMONIC DISTORTION
                                        ;*****************************************

000260                          CDIS.                           ;COMPUTE DISTORTION
                                                        ;*;FOR I=2,11; HARM(I)=0
000260  012700  000013                  MOV     #HARLIN,R0      ;ZERO HARMONIC POINTER ARRAY
000264  012701  000016                  MOV     #HARM+1,R1
000270  105021                  1$:     CLRB    (R1)+           ;SET ALL BYTES TO ZERO
000272  005300                          DEC     R0
000274  003375                          BGT     1$
                                                        ;*;DSTORT=-1
000276  172437  000704'                 LDF     @#M1,AC0        ;-1
000302  174075  000014                  STF     AC0,@14(R5)     ;DISTORTION = -1 = NOT COMPUTED
                                                        ;*;NHARM=LHARM
000306  017504  000016                  MOV     @16(R5),R4      ;NUMBER OF HARMONICS TO USE
000312  003552                          BLE     DONE            ;DON'T COMPUTE DISTORTION
                                                        ;*;IF (NHARM.GT.HARLIN) NHARM=HARLIN
000314  022704  000013                  CMP     #HARLIN,R4      ;LIMITED NUMBER
000320  002002                          BGE     5$              ;OKAY
000322  012704  000013                  MOV     #HARLIN,R4      ;USE MAXIMUM
000326  005304                  5$:     DEC     R4              ;USE AS COUNTER
                                                        ;*;IF (NHARM.LE.1) RETURN
000330  003543                          BLE     DONE            ;DON'T COMPUTE DISTORTION
                                                        ;*;PWR=0
000332  170401                          CLRF    AC1             ;PWR = 0 (INITIALIZE)
000334  174137  000000G                 STF     AC1,@#PWR
000340  012703  000001                  MOV     #1,R3           ;POINTER = L

                                                        ;*;DO 30 L=2,NHARM
000344  005203                  10$:    INC     R3              ;L = L+1 (START AT 2)
                                                        ;*;I=MOD(L*K,N)
000346  010301                          MOV     R3,R1           ;GET L
000350  070175  000006                  MUL     @6(R5),R1       ;L * NCYC
000354  006700                          SXT     R0
000356  071075  000004                  DIV     @4(R5),R0       ;L*NCYC / NSAM
000362  010137  000000G                 MOV     R1,@#I          ;=I
                                                        ;*;IF (MOD(I,N).EQ.0) IERR=-L
                                                        ;*;IF (MOD(I,N).EQ.0) GOTO 30
000366  006700                          SXT     R0
000370  071075  000004                  DIV     @4(R5),R0       ;I / NSAM
000374  005701                          TST     R1              ;IS REMAINDER 0?
000376  001005                          BNE     15$             ;NO
000400  010302                          MOV     R3,R2           ;GET L
000402  005402                          NEG     R2              ;-L
000404  010275  000020                  MOV     R2,@20(R5)      ;IERR=-L
000410  000466                          BR      40$
```

```
                                                                ;*;IF (I.LE.N/2) I=I
                                                                ;*;IF (I.GT.N/2) I=N-I
000412  013701  000000G     15$:    MOV     @#I,R1              ;RESTORE I TO R1
000416  017502  000004              MOV     @4(R5),R2           ;GET NSAM
000422  006202                      ASR     R2                  ;NSAM / 2
000424  020102                      CMP     R1,R2               ;COMPARE I AND NSAM/2
000426  003003                      BGT     20$                 ;I > NSAM/2
000430  010137  000000G             MOV     R1,@#I              ;I =< NSAM/2 SO KEEP I
000434  000405                      BR      30$
000436  167501  000004      20$:    SUB     @4(R5),R1           ;GET I - NSAM
000442  005401                      NEG     R1                  ;NSAM - I
000444  010137  000000G             MOV     R1,@#I              ;USE I = NSAM-I
                                                                ;*;DO 20 LL = 1,L
000450  012702  000001      30$:    MOV     #1,R2               ;LL=1
                                                                ;*;IF (HARM(LL).EQ.I) GOTO 30
000454  012700  177776G     35$:    MOV     #HARM-2,R0          ;HARM(-1) ADDRESS
000460  060200                      ADD     R2,R0               ;HARM(LL) ADDRESS
000462  060200                      ADD     R2,R0
000464  021001                      CMP     (R0),R1             ;IS IT I(=R1)?
000466  001437                      BEQ     40$                 ;IS USED
                                                                ;*;20 CONTINUE
000470  005202                      INC     R2                  ;LL=LL+1
000472  020203                      CMP     R2,R3               ;R3 IS STILL "L"
000474  002767                      BLT     35$                 ;L < LL
                                                                ;*;HARM(L) = I
000476  012700  177776G             MOV     #HARM-2,R0          ;HARM(-1) ADDRESS
000502  060300                      ADD     R3,R0               ;HARM(L) ADDRESS
000504  060300                      ADD     R3,R0
000506  010110                      MOV     R1,(R0)             ;=I

                                                                ;*;CALL DCRTZL (X,N,I,XR,XI)
000510  010546                      MOV     R5,-(SP)            ;SAVE R5
000512  010446                      MOV     R4,-(SP)            ;SAVE R4
000514  010346                      MOV     R3,-(SP)            ;SAVE R3
                                    .IF EQ PROH
                                    MOV     2(R5),@#DCARG+2     ;TRANSFER ADDRESS OF ARRAY
                                    .ENDC
000516  012705  000654'             MOV     #DCARG,R5           ;LOAD ARGUMENT LIST
000522  004737  000000G             JSR     PC,DCZL             ;DFT THE HARMONIC

                                                                ;*;PWR=PWR+2*(XI*XR+XI*XI)
000526  172475  000010              LDF     @10(R5),AC0         ;XR
000532  171000                      MULF    AC0,AC0             ;XR*XR
000534  172575  000012              LDF     @12(R5),AC1         ;XI
000540  171101                      MULF    AC1,AC1             ;XI*XI
000542  172001                      ADDF    AC1,AC0             ;XR*XR + XI*XI
000544  172000                      ADDF    AC0,AC0             ;2 (XR*XR + XI*XI)
000546  172537  000000G             LDF     @#PWR,AC1
000552  172100                      ADDF    AC0,AC1             ;SUM TO PWR
000554  174137  000000G             STF     AC1,@#PWR           ;AND SAVE

000560  012603                      MOV     (SP)+,R3            ;RESTORE R3
000562  012604                      MOV     (SP)+,R4            ;RESTORE R4
000564  012605                      MOV     (SP)+,R5            ;RESTORE R5
```

```
243                                                             ;*;30 CONTINUE
244 000566  005304                  40$:    DEC     R4                 ;DECREMENT COUNTER
245 000570  003365                          BGT     10$                ;NOT DONE--NEXT HARMONIC
246                                                             ;*;DSTORT=100*SQRT(PWR/(AMP*AMP))
247 000572  172475  000010                  LDF     @10(R5),AC0        ;GET VDFT
248 000576  170000                          CFCC                       ;CHECK FOR ZERO
249 000600  003413                          BLE     80$                ;ERROR!! (POTENTIAL PWR/0)
250 000602  171000                          MULF    AC0,AC0            ;VDFT * VDFT
251 000604  174560                          DIVF    AC0,AC1            ;PWR / (VDFT*VDFT)
252 000606  172401                          LDF     AC1,AC0            ;DISTORTION
253 000610  004767  000000G                 JSR     PC,@#SQRT
254 000614  172537  000674'                 LDF     @#CENT,AC1
255 000620  171001                          MULF    AC1,AC0            ;DISTORTION IN PERCENT
256 000622  174075  000014                  STF     AC0,@14(R5)        ;LOAD DISTORTION
257 000626  000404                          BR      DONE               ;DONE
258 000630  172437  000710'         80$:    LDF     @-M2,AC0           ;FOR ERROR CONDITION,
259 000634  174075  000014                  STF     AC0,@14(R5)        ;DISTORTION = -2
260
261 000640                          DONE:
262 000640  012604                          MOV     (SP)+,R4           ;RESTORE REGISTERS FROM STACK
263 000642  012603                          MOV     (SP)+,R3
264 000644  012602                          MOV     (SP)+,R2
265 000646  012601                          MOV     (SP)+,R1
266 000650  012600                          MOV     (SP)+,R0
267 000652  000207                          RTS     PC
268
269 000654  004[75  000000G 000000G 000000G DCARG:  5,RBUF,NSAN,I,XREAL,XIMAG
    000662  000000G 000000G 000000G
270 000670  041545  027346              ANGC:   .FLT2   57.2958 ;RADIANS-TO-DEGREES CONVERSION FACTOR
271 000674  041710  000000              CENT:   .FLT2   100.
272 000700  041664  000000              P90:    .FLT2   90.
273 000704  140200  000000              M1:     .FLT2   -1.
274 000710  140400  000000              M2:     .FLT2   -2.
275 000714  140600  000000              M3:     .FLT2   -3.
276 000720                          ARCBLK:
277 000720  000002                          .WORD   2
278 000722  000000G                         .WORD   XIMAG
279 000724  000000G                         .WORD   XREAL
280
281                                         .IF EQ PROM
282                                         ;RAM VARIABLES USED BY DAPD
283                                 XREAL:  .FLT2   0.0
284                                 XIMAG:  .FLT2   0.0
285                                 PWR:    .FLT2   0.0
286                                 PSIGN:  .WORD   0
287                                 I:      .WORD   0
288                                 NSAN:   .WORD   0
289                                 HARM:   .BLKW   HARLIM  ;ARRAY FILLED AS HARMONICS ARE USED
290                                 RBUF:   .WORD   0
291                                         .ENDC
292
293         000001                          .END
```

(5) MODULE DGZL - Subroutine to perform a discrete Fourier transform on a data sequence using Goertzel's algorithm.

Calling Sequence:

```
CALL DGZL     (X, N, K, XR, XI)
```

where the arguments are:

X	Floating-point data sequence
N	Length of sequence
K	Spectral line (number of cycles) to compute
XR	Real part of voltage
XI	Imaginary part of voltage

```
                                        .TITLE DGZL
                                        .IDENT /01/
                                        ;SUBROUTINE DGZL (X,N,K,XR,XI)
                                        ;REAL X(N)        !DATA SEQUENCE
                                        ;INTEGER N        !SEQUENCE LENGTH
                                        ;INTEGER K        !DFT SPECTRAL LINE NUMBER OF INTEREST
                                        ;                 !ALSO # CYCLES OF TEST SINUSOID IN N SAMPLES
                                        ;REAL XR          !REAL PART OF VOLTAGE
                                        ;REAL XI          !IMAGINARY PART OF VOLTAGE
                                        ;DGRTZL.FTN -- JD GEORGE (17-JUL-81) (ORIGINATOR)
                                        ;DGZL  .MAC -- RE SCOTT (02-OCT-81) (TRANSLATOR)
                                        ;GOERTZEL'S ALGORITHM
                                        ;TO COMPUTE K-TH LINE OF N POINT REAL SEQUENCE X(L)
                                        ;REFERENCE: G. GOERTZEL
                                        ;  "AN ALGORITM FOR THE EVALUATION OF FINITE TRIG SERIES"
                                        ;  AMERICAN MATHEMATICAL MONTHLY, V65, 1958, PP.34-35
                                        ;ALSO CHAPTER 24, PP.258-62
                                        ;  MATHEMATICAL METHODS FOR DIGITAL COMPUTERS VOL. 1
                                        ;  EDITED BY A. RALSTON & H.S. WILF
                                        ;  JOHN WILEY & SON, 1967
                                        ;A DOUBLE PRECISION GOERTZEL ROUTINE WHICH IS MORE ACCURATE AND
                                        ;FASTER THAN "SPDFT.FTN"
                                        ;  40% RUNNING TIME OF SPDFT.FTN
                                        ;  UP TO 512 POINTS AMPLITUDE ERROR < 0.0012%
                                        ;                  PHASE ERROR < 0.00001DEG
000000                                          AC0=%0                  ;FLOATING POINT ACCUMULATOR #0
000001                                          AC1=%1                  ;FLOATING POINT ACCUMULATOR #1
000002                                          ULP1=%2                 ;FLOATING POINT ACCUMULATOR #2
000003                                          ULP2=%3                 ;FLOATING POINT ACCUMULATOR #3
                                        ;SET PROM = 1 FOR RAM VARIABLES STORED EXTERNALLY
                                        ;SET PROM = 0 FOR RAM VARIABLES STORED INTERNALLY (NORMAL)
000001                                  PROM=1
000000  010546                 DGZL::  MOV     R5,-(SP)        ;PUSH THE ARGUMENT ADDRESS
000002  170011                         SETD                    ;SET FLOATING DOUBLE MODE
000004  170002                         SETI
000006  177075  000006                 LDCID   @6(R5),AC0      ;DFLOAT(K)
000012  177175  000004                 LDCID   @4(R5),AC1      ;DFLOAT(N)
000016  174401                         DIVD    AC1,AC0         ;A0=2.*PI*(K)/(N)
000020  171037  000204'                MULD    @#PI,AC0
000024  172000                         ADDD    AC0,AC0
000026  174037  000000G                STD     AC0,@#A0
000032  004767  000000G                JSR     PC,$$DCOS       ;C0=DCOS(A0)
000036  174037  000000G                STD     AC0,@#C0
000042  172000                         ADDD    AC0,AC0         ;C1=2.*C0
000044  174037  000000G                STD     AC0,@#C1
```

```
 58 000050  172437  000000G                 LDD     @#A0,AC0        ;S0=DSIN(A0)
 59 000054  004767  000000G                 JSR     PC,$$DSIN
 60 000060  174037  000000G                 STD     AC0,@#S0
 61
 62 000064  170402                          CLRD    ULP1            ;ULP1=0.0D0
 63 000066  170403                          CLRD    ULP2            ;ULP2=0.0D0
 64
 65 000070  012605                          MOV     (SP)+,R5        ;RESTORE ARGUMENT LIST
 66 000072  017501  000004                  MOV     @4(R5),R1       ;COUNTER N IN R1
 67 000076  010100                          MOV     R1,R0           ;DATA POINTER IN R0
 68 000100  005301                          DEC     R1              ;COUNT FROM N TO 2
 69 000102  006300                          ASL     R0
 70 000104  006300                          ASL     R0              ;REAL*4 DATA BUFFER
 71 000106  066500  000002                  ADD     2(R5),R0        ;ADDRESS OF LAST+1 DATA WORD
 72
 73                                                                 ;DO 10 L=N,2,-1
 74 000112  172437  000000G          10$:    LDD     @#C1,AC0        ;UL=X(L)+C1*ULP1-ULP2
 75 000116  171002                          MULD    ULP1,AC0
 76 000120  173003                          SUBD    ULP2,AC0
 77 000122  177540                          LDCFD   -(R0),AC1
 78 000124  172001                          ADDD    AC1,AC0
 79 000126  172702                          LDD     ULP1,ULP2       ;ULP2=ULP1
 80 000130  172600                          LDD     AC0,ULP1        ;ULP1=UL
 81 000132  077111                          SOB     R1,10$
 82
 83 000134  172437  000000G                 LDD     @#C0,AC0        ;XR=(X1+C0*ULP1-ULP2)/(N)
 84 000140  171002                          MULD    ULP1,AC0
 85 000142  173003                          SUBD    ULP2,AC0
 86 000144  177575  000002                  LDCFD   @2(R5),AC1
 87 000150  172001                          ADDD    AC1,AC0
 88 000152  177175  000004                  LDCID   @4(R5),AC1
 89 000156  174401                          DIVD    AC1,AC0
 90 000160  176075  000010                  STCDF   AC0,@10(R5)
 91
 92 000164  172437  000000G                 LDD     @#S0,AC0        ;XI=-S0*ULP1/(N)
 93 000170  171002                          MULD    ULP1,AC0
 94 000172  174401                          DIVD    AC1,AC0
 95 000174  170700                          NEGD    AC0
 96 000176  176075  000012                  STCDF   AC0,@12(R5)
 97
 98 000202  000207                          RTS     PC
 99 000204  040511  007732  121041  PI:     .FLT4   3.1415926535897932
    000212  064301
100                                         .IF EQ PROM
101                                 C0:     .FLT4   0.0
102                                 C1:     .FLT4   0.0
103                                 S0:     .FLT4   0.0
104                                 A0:     .FLT4   0.0
105                                         .ENDC
106         000001                          .END
```

(6) MODULE GPIOSR - This subroutine controls input and output along the IEEE-488 bus. There are three primary functions:

GPINI Initialize the interface and requisite variables; also set up address to use when group execute trigger (GET) is received.

Calling Sequence:

```
CALL GPINI      (RESET)
```

where the argument is:

RESET Address of routine to transfer to on GET

GPIN Listen function - read a byte string from the IEEE-488 bus, terminated by a character received with EOI true, or by input buffer overflow.

Calling Sequence:

```
CALL GPIN       (INBUF, MAXCH, NCH)
```

where the arguments are:

INBUF Byte buffer for input string

MAXCH Maximum number of characters allowed

NCH Number of characters received

GPOUT Talk function - set SRQ and transmit a byte string to the IEEE-488 bus.

Calling Sequence:

```
CALL GPOUT      (OUTBUF, NCH)
```

where the arguments are:

OUTBUF Byte buffer for output string

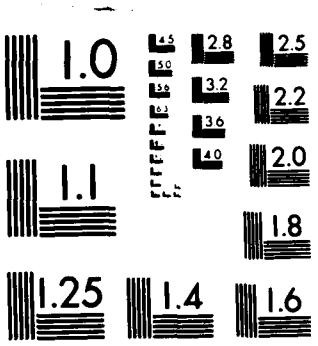
NCH Number of characters to output

Also included is an interrupt service routine. The only interrupt always enabled is the group execute trigger (GET) interrupt, which feature is used to effect a program abort or restart function. The bus-in and bus-out interrupts are enabled only when their respective subroutines are called and disabled on completion of transfer.

The stand-alone driver supplied by National Instruments Corp. was not used for two reasons: size (keeping the total program size under 4K) and to implement the special abort function. However, programming this complex interface was not a straightforward exercise.

AD-A126 980 A MICROCOMPUTER-BASED SAMPLING DIGITAL VOLTMETER(U) 2/2
NAVAL RESEARCH LAB WASHINGTON DC R E SCOTT ET AL.
31 MAR 83 NRL-MR-4872

UNCLASSIFIED F/G 14/2 NL



```
                                .TITLE  GPIOSR
                                .IDENT  /08/
                                ;
                                ;SUBROUTINES FOR HANDLING THE GPIBV-11 INTERFACE
                                ;    GPINI   -- INITIALIZE BUS
                                ;    GPIN    -- INPUT DATA FROM BUS (LISTEN)
                                ;    GPOUT   -- OUTPUT DATA TO BUS (TALK)
                                ;    ISRGP   -- INTERRUPT SERVICE ROUTINE
                                ;
                                ;AUTHOR: R. E. SCOTT, JR. (DECEMBER 1981)--V05
                                ;MODIFIED TO USE EOI ON READ (FEBRUARY 1982)--V06
                                ;MODIFIED TO USE R5 ARGUMENT LIST (FEBRUARY 1982)--V07
                                ;MODIFIED TO USE TRIGGER (NOT "AB") FOR ABORT--V08
                                ;NRL/UNDERWATER SOUND REFERENCE DETACHMENT

                                ;INITIALIZE--CALL GPINI (RESET)
                                ;INITIALIZES THE GPIBV-11 INTERFACE,
                                ;LOADS THE INTERRUPT VECTOR AND PSW (IF ALLOWED),
                                ;LOADS THE "RESET" ADDRESS (GO TO IF "CET" RECEIVED),
                                ;AND ENABLES BOTH INPUT AND OUTPUT INTERRUPTS.
                                ;
                                ;INPUT--CALL GPIN (INBUF,MAXCH,NCH)
                                ;ALLOWS A STRING OF VARIABLE LENGTH "NCH" (BUT LESS THAN
                                ;"MAXCH") TO BE LOADED IN "INBUF". EOI MUST COME WITH LAST BYTE.
                                ;
                                ;OUTPUT--CALL GPOUT (OUTBUF,NCH)
                                ;OUTPUTS A STRING "OUTBUF" OF VARIABLE LENGTH "NCH"
                                ;TO THE GPIB BUS.
                                ;
                                ;INTERRUPT SERVICE ROUTINE--
                                ;CHECKS FOR INPUT OR OUTPUT INTERRUPTS AND HANDLES WHICHEVER
                                ;AS APPROPRIATE.  IT ALSO CHECKS FOR ONE SPECIAL INPUT, THE
                                ;RESET (ABORT) FUNCTION, WHICH REINITIALIZES EVERYTHING.
                                ;
                                ;GPIB REGISTERS REQUIRED:
000270                          GPVEC  =    270 ;INTERRUPT VECTOR ADDRESS
164000                          GPISR  = 164000 ;INTERRUPT SERVICE REGISTER (RO)
164000                          GPIMR  = 164000 ;INTERRUPT MASK REGISTER (WO)
164001                          GPCTSR = 164001 ;CONTROL STATUS REGISTER (RO)
164002                          GPCSR  = 164002 ;COMMAND STATUS REGISTER (RO)
164004                          GPAMR  = 164004 ;ADDRESS MODE REGISTER (WO)
164006                          GPACR  = 164006 ;AUXILIARY CONTROL REGISTER (RW)
164010                          GPASWR = 164010 ;ADDRESS SWITCH REGISTER (RO)
164010                          GPADR  = 164010 ;ADDRESS REGISTER (WO)
164012                          GPSPR  = 164012 ;SERIAL POLL REGISTER (RW)
164015                          GPCCR  = 164015 ;CONTROLLER COMMAND REGISTER (WO)
164016                          GPDIR  = 164016 ;DATA IN REGISTER (RO)
164016                          GPDOR  = 164016 ;DATA OUT REGISTER (WO)

000001                          PROM=1  ;SET TO 1 WHEN INTERRUPT VECTORS LOADED INTO PROM;
                                        ;OTHERWISE THEY ARE LOADED DYNAMICALLY BY GPINI.
000000                          RAM=0   ;ALSO, NO LOCAL RAM VARIABLES IF PROM=1.
```

```
                                                ;***** GPIB INITIALIZATION SUBROUTINE *****
000000                                  GPINI::
000000  010046                                  MOV     R0,-(SP)        ;SAVE REGISTER 0 ON STACK
000002  000005                                  RESET                   ;*??NEEDED??
000004  105037  164006                          CLRB    @#GPACR         ;INITIALIZE GPIB AUX REGISTER
                                        ;       MOVB    #10,@#GPAMR     ;SET HOLD RFD ON EOI
000010  113700  164010                          MOVB    @#GPASWR,R0     ;GET CONTROLLER ADDRESS
000014  032700  000140                          BIT     #140,R0         ;TEST CERTAIN SWITCHES ON CONTROLLER
000020  001401                                  BEQ     5$              ;OKAY; CONTINUE
000022  000000                                  HALT                    ;ERROR: SAC AND/OR EXT SET
000024  042700  177740                  5$:     BIC     #177740,R0      ;ENABLE TALKING AND LISTENING
000030  110037  164010                          MOVB    R0,@#GPADR      ;LOAD CONTROLLER ADDRESS
                                                .IF EQ PROM             ;DON'T DO IF VECTOR SPACE IS ROM
                                                MOV     #ISRGP,@#GPVEC  ;LOAD INTERRUPT VECTOR
                                                MOV     #340,@#GPVEC+2  ;AND PSW
                                                .ENDC
000034  012600                                  MOV     (SP)+,R0        ;RESTORE REGISTER 0
000036  016537  000002  000000G                 MOV     2(R5),@#CTRIG   ;LOAD ADDRESS TO GO IF TRIGGER
000044  005037  000000G                         CLR     @#GPEOF         ;CLEAR EOI FLAG
000050  112737  000240  164000                  MOVB    #240,@#GPIMR    ;ENABLE GET INTERRUPTS
000056  112737  000010  164015                  MOVB    #10,@#GPCCR     ;ENABLE MASTER INTERRUPT
000064  000207                                  RTS     PC              ;BUS IS NOW INITIALIZED

                                                ;***** GPIB STRING INPUT SUBROUTINE *****
000066                                  GPIN::
                                        ;       BISB    #100,@#GPACR    ;RELEASE RFD HANDSHAKE FROM LAST INPUT
000066  016537  000002  000000G                 MOV     2(R5),@#GPBIN   ;INPUT BUFFER ADDRESS
000074  017537  000004  000000G                 MOV     @4(R5),@#GPMAX  ;MAXIMUM LENGTH ALLOWED
000102  005037  000000G                         CLR     @#GPNCH         ;NUMBER OF CHARACTERS RECEIVED
000106  112737  000241  164000                  MOVB    #241,@#GPIMR    ;ENABLE BI INTERRUPTS
000114  000001                          10$:    WAIT                    ;WAIT FOR "ANY" INTERRUPT
000116  005737  000000G                         TST     @#GPEOF         ;END-OF-INPUT?
000122  001004                                  BNE     20$             ;YES
000124  023737  000000G 000000G                 CMP     @#GPMAX,@#GPNCH ;BUFFER FULL?
000132  003370                                  BGT     10$             ;NOT DONE/NOT FULL

000134  112737  000240  164000          20$:    MOVB    #240,@#GPIMR    ;DISABLE BI INTERRUPTS
000142  005037  000000G                         CLR     @#GPEOF
000146  013775  000000G 000006                 MOV     @#GPNCH,@6(R5)  ;RETURN NUMBER OF CHARACTERS
000154  000207                                  RTS     PC              ;INPUT COMPLETE


                                                ;***** GPIB STRING OUTPUT SUBROUTINE *****
000156                                  GPOUT::
                                        ;       BISB    #100,@#GPACR    ;RELEASE RFD HANDSHAKE FROM LAST INPUT
000156  016537  000002  000000G                 MOV     2(R5),@#GPBOUT  ;GET OUTPUT BUFFER ADDRESS
000164  017537  000004  000000G                 MOV     @4(R5),@#GPNCH  ;GET NUMBER BYTES TO OUTPUT
000172  112737  000340  164000                  MOVB    #340,@#GPIMR    ;ENABLE BO INTERRUPTS
000200  112737  000102  164012                  MOVB    #102,@#GPSPR    ;SET SRQ AND READ BIT
000206  000001                          10$:    WAIT                    ;WAIT FOR "ANY" INTERRUPT
000210  005737  000000G                         TST     @#GPNCH         ;ALL CHARACTERS OUTPUT?
000214  003374                                  BGT     10$             ;NO--WAIT MORE
000216  112737  000240  164000                  MOVB    #240,@#GPIMR    ;DISABLE BO INTERRUPTS
000224  105037  164012                          CLRB    @#GPSPR         ;** HOST DOESN'T CLEAR "2"
000230  000207                                  RTS     PC              ;OUTPUT COMPLETE
```

```
                                                ;***** GPIB I/O INTERRUPT SERVICE ROUTINE *****
000232                                  ISRGP::
000232  032737  001000  164000                  BIT     #1000,@#GPISR   ;;;TEST IFC BIT
000240  001407                                 BEQ     20$             ;;;OKAY IF NOT SET
000242  142737  000002  164015  10$:           BICB    #2,@#GPCCR      ;;;CLEAR IFC BIT
000250  132737  000002  164015                  BITB    #2,@#GPCCR      ;;;SUCCESSFUL?
000256  001371                                 BNE     10$             ;;;NO--TRY AGAIN
                                                ;
000260  132737  000001  164000  20$:           BITB    #1,@#GPISR      ;;;"BI" INTERRUPT SET?
000266  001013                                 BNE     40$             ;;;YES
000270  132737  000100  164000                  BITB    #100,@#GPISR    ;;;"BO" INTERRUPT SET?
000276  001026                                 BNE     50$             ;;;YES
000300  132737  000040  164000                  BITB    #40,@#GPISR     ;;;"GET" INTERRUPT SET?
000306  001001                                 BNE     30$             ;;;YES
000310  000436                                 BR      100$            ;;;INTERRUPT ERROR; TRY AGAIN
                                                ;
000312  000177  000000G         30$:           JMP     @GTRIG          ;;;RESTART
                                                ;
                                                ;;;IF DATA RECEIVED FROM BUS
000316  132737  000002  164000  40$:           BITB    #2,@#GPISR      ;;;END-OF-INPUT?
000324  001403                                 BEQ     41$             ;;;NO--CONTINUE
000326  012737  000001  000000G                 MOV     #1,@#GPEOF      ;;;SET EOI FLAG
000334  113777  164016  000000G 41$:           MOVB    @#GPDIR,@GPBIN  ;;;GET A CHARACTER
000342  005237  000000G                         INC     @#GPBIN         ;;;INCREMENT INPUT ADDRESS
000346  005237  000000G                         INC     @#GPNCH         ;;;INCREMENT CHARACTER COUNT
000352  000415                                 BR      100$
                                                ;
                                                ;;;IF DATA TO SEND TO BUS
000354  117737  000000G  164016  50$:           MOVB    @GPBOUT,@#GPDOR ;;;SEND OUT DATA BYTE
000362  005237  000000G                         INC     @#GPBOUT        ;;;INCREMENT OUTPUT ADDRESS
000366  005337  000000G                         DEC     @#GPNCH         ;;;DONE WITH STRING?
000372  003401                                 BLE     60$             ;;;YES--CLEANUP
000374  000404                                 BR      100$            ;;;NO--NEXT?
000376  132737  000100  164000  60$:           BITB    #100,@#GPISR    ;;;DATA-OUT REGISTER CLEAR?
000404  001774                                 BEQ     60$             ;;;NO
                                                ;
000406  000002                         100$:   RTI
                                                ;
                                                .IF NE RAM      ;ASSEMBLE IF VARIABLES NOT PROM
                                        GPEOF:  .WORD   0       ;E-O-I FOUND? FLAG
                                        GPBIN:  .WORD   0       ;INPUT BUFFER ADDRESS
                                        GPBOUT: .WORD   0       ;OUTPUT BUFFER ADDRESS
                                        GPMAX:  .WORD   0       ;MAXIMUM CHARACTERS ON INPUT
                                        GPNCH:  .WORD   0       ;CHARACTER COUNT FOR INPUT OR OUTPUT
                                        GTRIG:  .WORD   0       ;ADDRESS TO GO IF "GET" RECEIVED
                                                .ENDC
000001                                          .END
```

(7) MODULE GRMS - This subroutine computes parameters from statistical techniques.

Calling Sequence:

CALL GRMS (N, X, RMS, DC, PEAKMX, PEAKMN)

where the arguments are:

N	Length of sequence
X	Floating-point data sequence
RMS	Returned rms voltage (standard deviation)
DC	Returned dc voltage (average)
PEAKMX	Peak voltage (maximum)
PEAKMN	Peak voltage (minimum)

NOTE: This code was originally written by R.W. Anderson and subsequently modified by R.W. Luckey, both of NRL-USRD.

```
                                        .TITLE CRMS
                                        .IDENT  /02/

                                        ;SUBROUTINE TO COMPUTE TIME-AVERAGE VOLTAGES: RMS, DC,
                                        ;MAXIMUM, AND MINIMUM.
                                        ;
                                        ;AUTHOR: RW ANDERSON (SOMETIME IN 1980)
                                        ;MODIFIED: RW LUCKEY (FOR BOTH PEAKS; 1981)

                                        ;FORTRAN CALL--
                                        ;CALL CRMS (NPTS,VALS,RMS,AVG,PEAKMX,PEAKMN)

                                        ;AC0 - IS THE SUMMING REGISTER
                                        ;AC1 - IS THE REGISTER FOR THE SUM OF THE SQUARES
                                        ;AC2 -
                                        ;AC3 - NUMBER OF POINTS
                                        ;AC4 - PEAK MINIMUM - STORAGE - REGISTER NOT GENERALLY USABLE
                                        ;AC5 - PEAK MAXIMUM - STORAGE - REGISTER NOT GENERALLY USABLE

                                        ;R0 - LOOP COUNTER - INITIALIZED TO NPTS
                                        ;R1 - ADDRESS OF THE VALUES

                 000000                         AC0 = %0
                 000001                         AC1 = %1
                 000002                         AC2 = %2
                 000003                         AC3 = %3
                 000004                         AC4 = %4
                 000005                         AC5 = %5

000000  170001                  CRMS::  SETF
000002  172475  000004                  LDF     @4(R5),     AC0         ; PEAK MINIMUM = X(1)
000006  174004                          STF     AC0,        AC4         ; STORE IN AC4
000010  172475  000004                  LDF     @4(R5),     AC0         ; PEAK MAXIMUM = X(1)
000014  174005                          STF     AC0,        AC5         ; STORE IT IN AC5
000016  017500  000002                  MOV     @2(R5),     R0          ; SET UP LOOP COUNTER
000022  177300                          LDCIF   R0,         AC3         ; NUMBER OF POINTS (NPTS)
000024  016501  000004                  MOV     4(R5),      R1          ; ADDRESS OF DATA IN REGISTER ONE
000030  170400                          CLRF    AC0                     ; ZERO SUMMING REGISTER
000032  170401                          CLRF    AC1                     ; ZERO SQUARES SUMMING REGISTER
000034  172611                  LOOP:   LDF     (R1),       AC2         ; GET A VALUE
000036  172002                          ADDF    AC2,        AC0         ; ADD IT TO SUMMING REGISTER
000040  171202                          MULF    AC2,        AC2         ; SQUARE IT
000042  172102                          ADDF    AC2,        AC1         ; ADD IT TO SQUARES SUMMING REGISTER
000044  172621                          LDF     (R1)+,      AC2         ; GET THE VALUE AGAIN
000046  173605                          CMPF    AC5,        AC2         ; SEE IF NEW MAX PEAK
000050  170000                          CFCC                            ; COPY FUNCTION CODES
000052  100001                          BPL     MIN                     ; BRANCH IF NOT NEW MAX PEAK
000054  174205                          STF     AC2,        AC5         ; SAVE NEW MAX PEAK
000056  173604                  MIN:    CMPF    AC4,        AC2         ; SEE IF NEW MIN PEAK
000060  170000                          CFCC                            ; COPY FUNCTION CODES
000062  100401                          BMI     CONT                    ; BRANCH IF NOT NEW MIN PEAK
000064  174204                          STF     AC2,        AC4         ; SAVE NEW MIN PEAK
000066  077016                  CONT:   SOB     R0,         LOOP        ; LOOP IF NOT FINISHED
```

```
55 000070  174403                DIVF  AC3,    AC0          ; AVERAGE (AVG) = SUM / NPTS
56 000072  174075  000010        STF   AC0,    @10(R5)      ; STORE AVERAGE (AVG) - DC
57 000076  172605                LDF   AC5,    AC2          ; MOVE MAX PEAK INTO AC2
58 000100  173200                SUBF  AC0,    AC2          ; PEAK MAX = PEAK MAX - AVG (DC)
59 000102  174275  000012        STF   AC2,    @12(R5)      ; STORE PEAK MAX
60 000106  172604                LDF   AC4,    AC2          ; MOVE MIN PEAK INTO AC2
61 000110  173200                SUBF  AC0,    AC2          ; PEAK MIN = PEAK MIN - AVG (DC)
62 000112  174275  000014        STF   AC2,    @14(R5)      ; STORE PEAK MIN
63 000116  171000                MULF  AC0,    AC0          ; SQUARE THE AVERAGE (DC)
64 000120  174503                DIVF  AC3,    AC1          ; SQUARE AVERAGE = SUMSQ / NPTS
65 000122  173100                SUBF  AC0,    AC1          ; SUBTRACT AVERAGE SQUARED (DC)**2
66 000124  174100                STF   AC1,    AC0          ; PUT IT IN AC0
67 000126  004767  00000C        JSR   PC,     $$SQRT       ; GET THE SQUARE ROOT
68 000132  174075  000006        STF   AC0,    @6(R5)       ; STORE RMS
69 000136  000207                RTS   PC                   ; RETURN FROM SUBROUTINE
70         000001                .END
```

(8) LSIVAR - The following is an alphabetical list (with definitions) of the variables used in the software. All are used by LSIVM except for those used by the subroutines noted in parentheses.

AO	Exponent factor (DGZL)
AD1C	Input port for channel 1
AD2C	Input port for channel 2
AD3C	Input port for channel 3
ADBUF	A/D buffers (three contiguous arrays)
ADS	Numbers of A/D's used
ADST	Measurement status register (A32D)
ADT	DMA to interrupt on (ignore all others)
ADUM	A/D pseudo-register (A32D)
ADWL	Length in bytes of buffer to send to host
AERRS	Automatic retries when A/D errors occur
AVGCOD	Averaging mode (0 = don't, or cycles per averaged subsequence)
BSIZE	Buffer size allowed per channel
BUF1	12-bit integer values from A/D #1
BUF2	12-bit integer values from A/D #2
BUF3	12-bit integer values from A/D #3
CO	Cosine factor (DGZL)
C1	Cosine factor (DGZL)
CFACT	Current factor (I = CFACT*E)
CHERRY	Display character buffer
CHNO	Channel number being computed
COMTYP	Computation type (not used)
COV	Covariance (ac power)
DATA	Start of LSIVM input parameter buffer

DCV	Average (dc) voltage from statistical computation
DFD	DFT data array
DIST1	Distortion of channel 1
DIST2	Distortion of channel 2
DIST3	Distortion of channel 3
DIST	Harmonic distortion
DLUN	Display device (1 = terminal, 2 = display)
DSTYLE	Display style (0-5)
ECNT	Counter for A/D error retry
FSAM	Sample frequency (Hz)
FSIG	Signal frequency (Hz)
GAIN1	A/D gain channel 1 (1, 2, 5, 10)
GAIN2	A/D gain channel 2 (1, 2, 5, 10)
GAIN3	A/D gain channel 3 (1, 2, 5, 10)
GAIN	A/D converter gains array
GATED	Gated voltage flag (0 = no, 1 = yes)
GPBIN	Input buffer address (GPIOSR)
GPBOUT	Output buffer address (GPIOSR)
GPEOF	E-O-I found - flag (0 = no, 1 = yes) (GPIOSR)
GPMAX	Maximum characters on input (GPIOSR)
GPNCH	Character count for input or output (GPIOSR)
GTRIG	Address to go to if "GET" received (GPIOSR)
HAR	Number of harmonics to use to compute distortion
HARM	Harmonics - used flag array (DAPD)
HTC	Specific harmonic (line + 1) to compute
I	Line number (DAPD)
IBLEN	Bytes received from IEEE-488 bus

IC	Command byte 1
ICA	Count of A/D conversions
ICE	Count of A/D errors
ICN	Command byte 2 (may be channel number)
IERD	Error status of DFT
INBUF	IEEE-488 bus input packet
IP	Command parameter string
ISW	LSIVM status word
IWD	What DMA to interrupt from (A32D)
LADD	Pointer to displayed parameters
LCTR	Counter for displayed parameters
LDAT	Parameter to display
LPTR	Index for displayed parameters
NCAV	Sequences to average (NCYC/AVGCOD)
NCYC	Cycles per sequence
NCYF	Cycles to compute (AVGCOD); averaging changes
NCYHA	Cycles to compute (for a harmonic)
NSAM	Samples per sequence (DAPD)
PAD	Samples to convert per DMA transfer per channel
PDFT	Phase from DFT
PEKVM	Peak (-) voltage from inspection
PEKVP	Peak (+) voltage from inspection
PHAS1	Phase of channel 1
PHAS2	Phase of channel 2
PHAS3	Phase of channel 3
POWER	Total power (ac and dc)
PSIGN	Phase sign (DAPD)

PTC	Points to compute (PTR/NCYC); averaging changes this
PTR	Samples to transfer to host (samples per sequence)
PWR	Harmonic power (DAPD)
RAA	Argument list for writing buffer to host
RAM	Start of program RAM space
RAN	Address of buffer to send to host
RBUF	One voltage buffer (converted to floating-point)
RMD	Statistical data array
RMSV1	RMS voltage in channel 1
RMSV2	RMS voltage in channel 2
RMSV3	RMS voltage in channel 3
RMSV	RMS voltage from statistical analysis
RPHAS	Phase difference between channels 1 and 2
SO	Sine factor (DGZL)
SJM	Second joint moment (total power)
STACK	Start (top) of program stack
STPT	Number of first sample to transfer to host
STRING	Used by print
T1	DMA #1 pseudo-register (A32D)
T2	DMA #2 pseudo-register (A32D)
T3	DMA #3 pseudo-register (A32D)
VDFT	Total rms voltage from DFT
WAD	What A/D converters used (A32D)
WAV	Average window magnitude
WIND	Window buffer (floating-point)
WMODE	Window mode (0 = no, 1 = yes)
XIMAG	Imaginary voltage part (DAPD)

XREAL Real voltage part (DAP)

```
                                        .TITLE  LSIVAR
                                        .IDENT  /06/

                                        ;GLOBAL VARIABLES (DYNAMIC) FOR LSIVM


000000                                  .PSECT $$VARL,REL,RO    ;NOT REALLY RO!! TO FOOL TASKBUILDER!!
000000                                  .BLKB   2452    ;SHIFT BEGINNING OF RAM TO 4K BOUNDARY

        002000                  BSIZE==1024.            ;BUFFER SIZE ALLOWED PER CHANNEL

002452                          RAM::   .BLKB   1000    ;START OF PROGRAM RAM SPACE
003452  000000                  STACK:: .WORD   0       ;START (TOP) OF PROGRAM STACK

003454  000000                  ICA::   .WORD   0       ;COUNT OF A/D CONVERSIONS
003456  000000                  ICE::   .WORD   0       ;COUNT OF A/D ERRORS

003460                          INBUF::                 ;IEEE-488 BUS INPUT PACKET
003460      000                 IC::    .BYTE   0       ;COMMAND BYTE 1
003461      000                 ICN::   .BYTE   0       ;COMMAND BYTE 2 (MAYBE CHANNEL NUMBER)
003462                          IP::    .BLKW   40.     ;COMMAND PARAMETER STRING
003602  000000                  ISW::   .WORD   0       ;LSIVM STATUS WORD
003604  000000                  IBLEN:: .WORD   0       ;BYTES RECEIVED FROM IEEE-488 BUS
003606  000000                  CHNO::  .WORD   0       ;CHANNEL NUMBER BEING COMPUTED

003610  000000                  GPEOF:: .WORD   0       ;E-O-I FOUND? FLAG (0=NO; 1=YES) (GPIOSR)
003612  000000                  GPBIN:: .WORD   0       ;INPUT BUFFER ADDRESS (GPIOSR)
003614  000000                  GPBOUT::.WORD   0       ;OUTPUT BUFFER ADDRESS (GPIOSR)
003616  000000                  GPMAX:: .WORD   0       ;MAXIMUM CHARACTERS ON INPUT (GPIOSR)
003620  000000                  GPNCH:: .WORD   0       ;CHARACTER COUNT FOR INPUT OR OUTPUT (GPIOSR)
003622  000000                  GTRIC:: .WORD   0       ;ADDRESS TO GO TO IF "GET" RECEIVED (GPIOSR)

003624                          DATA::                  ;START OF LSIVM INPUT PARAMETER BUFFER
003624                          CAIN::                  ;START OF A/D CONVERTER GAINS ARRAY
003624  000000                  CAIN1:: .WORD   0       ;A/D GAIN CHANNEL 1 (1,2,5,10)
003626  000000                  CAIN2:: .WORD   0       ;A/D GAIN CHANNEL 2 (1,2,5,10)
003630  000000                  CAIN3:: .WORD   0       ;A/D GAIN CHANNEL 3 (1,2,5,10)
003632  000000                  PTR::   .WORD   0       ;SAMPLES TO TRANSFER TO HOST (SAMPLES PER SEQUENCE)
003634  000000                  STPT::  .WORD   0       ;NUMBER OF FIRST SAMPLE TO TRANSFER TO HOST
003636  000000                  ADS::   .WORD   0       ;NUMBERS OF A/D'S USED (1=#1,2=#2,4=#3; 7=ALL)
003640  000000                  ADT::   .WORD   0       ;DMA TO INTERRUPT ON (IGNORE ALL OTHERS)
003642  000000                  PAD::   .WORD   0       ;SAMPLES TO CONVERT PER DMA TRANSFER PER CHANNEL
003644  000000                  NCYC::  .WORD   0       ;CYCLES PER SEQUENCE
003646  000000                  HAR::   .WORD   0       ;NUMBER OF HARMONICS TO USE TO COMPUTE DISTORTION
003650  000000                  AVCCOD::.WORD   0       ;AVERAGING MODE (0=DON'T; OR CYCLES PER AVERAGE SUBSEQ.)
003652  000000                  DSTYLE::.WORD   0       ;DISPLAY STYLE (0-5)
003654  000000                  COMTYP::.WORD   0       ;COMPUTATION TYPE (NOT USED)
003656  000000                  AERRS:: .WORD   0       ;AUTOMATIC RETRIES WHEN A/D ERRORS OCCUR
003660  000000                  GATED:: .WORD   0       ;GATED VOLTAGE FLAG (0=NO; 1=YES)
003662  000000                  WMODE:: .WORD   0       ;WINDOW MODE (0=NO; 1=YES)
003664  000000                  DLUN::  .WORD   0       ;DISPLAY DEVICE (1=TERMINAL; 2=DISPLAY)
003666  000000                  HTC::   .WORD   0       ;SPECIFIC HARMONIC (LINE+1) TO COMPUTE
003670  000000                  AD1C::  .WORD   0       ;INPUT PORT FOR CHANNEL 1
003672  000000                  AD2C::  .WORD   0       ;INPUT PORT FOR CHANNEL 2
003674  000000                  AD3C::  .WORD   0       ;INPUT PORT FOR CHANNEL 3
```

```
 57 003676  000000  000000          FSIG::  .FLT2   0.0     ;SIGNAL FREQUENCY (HZ)
 58 003702  000000  000000          FSAM::  .FLT2   0.0     ;SAMPLE FREQUENCY (HZ)
 59
 60 003706  000000                  PTC::   .WORD   0       ;POINTS TO COMPUTE (PTR/NCYC); AVERAGING CHANGES
 61 003710  000000                  NCYF::  .WORD   0       ;CYCLES TO COMPUTE (AVGCOD); AVERAGING CHANGES
 62 003712  000000                  NCYHA:: .WORD   0       ;CYCLES TO COMPUTE (FOR A HARMONIC)
 63 003714  000000                  NCAV::  .WORD   0       ;SEQUENCES TO AVERAGE (NCYC/AVGCOD)
 64 003716  000000                  ECNT::  .WORD   0       ;COUNTER FOR A/D ERROR RETRY
 65 003720  000000                  ADWL::  .WORD   0       ;LENGTH IN BYTES OF BUFFER TO SEND TO HOST
 66
 67 003722                          DFD::                   ;START OF DFT DATA ARRAY
 68 003722  000000  000000          VDFT::  .FLT2   0.0     ;TOTAL RMS VOLTAGE FROM DFT
 69 003726  000000  000000          PDFT::  .FLT2   0.0     ;PHASE FROM DFT
 70 003732  000000  000000          DIST::  .FLT2   0.0     ;HARMONIC DISTORTION
 71
 72 003736                          RMD::                   ;START OF TIME-INTEGRATION DATA ARRAY
 73 003736  000000  000000          RMSV::  .FLT2   0.0     ;RMS VOLTAGE FROM TIME INTEGRATION
 74 003742  000000  000000          PEKVP:: .FLT2   0.0     ;PEAK(+) VOLTAGE FROM INSPECTION
 75 003746  000000  000000          PEKVM:: .FLT2   0.0     ;PEAK(-) VOLTAGE FROM INSPECTION
 76 003752  000000  000000          DCV::   .FLT2   0.0     ;AVERAGE (DC) VOLTAGE FROM TIME INTEGRATION
 77
 78 003756  000000  000000          POWER:: .FLT2   0.0     ;TOTAL POWER (SUM OF E*I)
 79 003762  000000  000000          CFACT:: .FLT2   0.0     ;CURRENT FACTOR (I=CFACT*E)
 80
 81 003766  000000  000000          SJM::   .FLT2   0.0     ;SECOND JOINT MOMENT
 82 003772  000000  000000          COV::   .FLT2   0.0     ;COVARIANCE
 83
 84 003776  000000  000000  000000  RAA::   0,0,0           ;ARGUMENT LIST FOR WRITING BUFFER TO HOST
 85 004004  000000                  RAN::   .WORD   0       ;ADDRESS OF BUFFER TO SEND TO HOST
 86                                                         ;RAM VARIABLES USED BY A32D
 87 004006  000000                  WAD::   .WORD   0       ;WHAT A/D CONVERTERS USED (A32D)
 88 004010  000000                  IWD::   .WORD   0       ;WHAT DMA TO INTERRUPT FROM (A32D)
 89 004012  000000                  T1::    .WORD   0       ;DMA #1 PSEUDO-REGISTER (A32D)
 90 004014  000000                  T2::    .WORD   0       ;DMA #2 PSEUDO-REGISTER (A32D)
 91 004016  000000                  T3::    .WORD   0       ;DMA #3 PSEUDO-REGISTER (A32D)
 92 004020  000000                  ADUM::  .WORD   0       ;A/D PSEUDO REGISTER (A32D)
 93 004022  000000                  ADST::  .WORD   0       ;MEASUREMENT STATUS REGISTER (A32D)
 94 004024  000000                  ADD::   .WORD   0       ;CHANNEL 1 PORT ADDRESS * 400
 95 004026  000000                  BDD::   .WORD   0       ;CHANNEL 2 PORT ADDRESS * 400
 96 004030  000000                  CDD::   .WORD   0       ;CHANNEL 3 PORT ADDRESS * 400
 97
 98 004032  000000                  LCTR::  .WORD   0       ;COUNTER FOR DISPLAYED PARAMETERS
 99 004034  000000                  LPTR::  .WORD   0       ;INDEX FOR DISPLAYED PARAMETERS
100 004036  000000                  LADD::  .WORD   0       ;POINTER TO DISPLAYED PARAMETERS
101 004040  000000                  LDAT::  .WORD   0       ;PARAMETER TO DISPLAY
102 004042                          STRING::.BLKW   6       ;USED BY PRINT
103
```

```
                                                        ;RAM VARIABLES USED BY DAPD
004056  000000  000000        XREAL:: .FLT2   0.0      ;REAL VOLTAGE PART (DAPD)
004062  000000  000000        XIMAG:: .FLT2   0.0      ;IMAGINARY VOLTAGE PART (DAPD)
004066  000000  000000        PWR::   .FLT2   0.0      ;HARMONIC POWER (DAPD)
004072  000000                PSIGN:: .WORD   0        ;PHASE SIGN (DAPD)
004074  000000                I::     .WORD   0        ;LINE NUMBER (DAPD)
004076  000000                NSAM::  .WORD   0        ;SAMPLES PER SEQUENCE (DAPD)
004100                        HARM::  .BLKW   20.      ;HARMONICS-USED FLAG ARRAY (DAPD)

004150  000000                IERD::  .WORD   0        ;ERROR STATUS OF DFT
```



```
                                                        ;RAM VARIABLES USED BY DGZL
004152  000000  000000  000000  A0::    .FLT4   0.0      ;FACTOR (DGZL)
004160  000000
004162  000000  000000  000000  C0::    .FLT4   0.0      ;COSINE FACTOR (DGZL)
004170  000000
004172  000000  000000  000000  C1::    .FLT4   0.0      ;COSINE FACTOR (DGZL)
004200  000000
004202  000000  000000  000000  S0::    .FLT4   0.0      ;SINE FACTOR (DGZL)
004210  000000

004212                        CHERRY::.BLKB   36.      ;CHERRY DISPLAY CHARACTER BUFFER
004256  000000  000000        RMSV1:: .FLT2   0.0      ;RMS VOLTAGE IN CHANNEL 1
004262  000000  000000        RMSV2:: .FLT2   0.0      ;RMS VOLTAGE IN CHANNEL 2
004266  000000  000000        RMSV3:: .FLT2   0.0      ;RMS VOLTAGE IN CHANNEL 3
004272  000000  000000        PHAS1:: .FLT2   0.0      ;PHASE OF CHANNEL 1
004276  000000  000000        PHAS2:: .FLT2   0.0      ;PHASE OF CHANNEL 2
004302  000000  000000        PHAS3:: .FLT2   0.0      ;PHASE OF CHANNEL 3
004306  000000  000000        DIST1:: .FLT2   0.0      ;DISTORTION OF CHANNEL 1
004312  000000  000000        DIST2:: .FLT2   0.0      ;DISTORTION OF CHANNEL 2
004316  000000  000000        DIST3:: .FLT2   0.0      ;DISTORTION OF CHANNEL 3
004322  000000  000000        RPHAS:: .FLT2   0.0      ;PHASE DIFFERENCE BETWEEN CHANNELS 1 AND 2

004326                        ADBUF::                  ;A/D BUFFERS (THREE CONTIGUOUS ARRAYS)
004326                        BUF1::  .BLKW   BSIZE    ;12-BIT INTEGER VALUES FROM A/D #1
010326                        BUF2::  .BLKW   BSIZE    ;12-BIT INTEGER VALUES FROM A/D #2
014326                        BUF3::  .BLKW   BSIZE    ;12-BIT INTEGER VALUES FROM A/D #3

020326                        RBUF::  .BLKW   2*BSIZE  ;ONE VOLTAGE BUFFER (CONVERTED TO FLOATING-POINT)
030326  000000  000000        WAV::   .FLT2   0.0      ;AVERAGE WINDOW MAGNITUDE
030332                        WIND::  .BLKW   2*BSIZE  ;WINDOW BUFFER (FLOATING-POINT)

        000001                          .END
```

(BLANK PAGE)

APPENDIX F

Voltmeter Memory Map

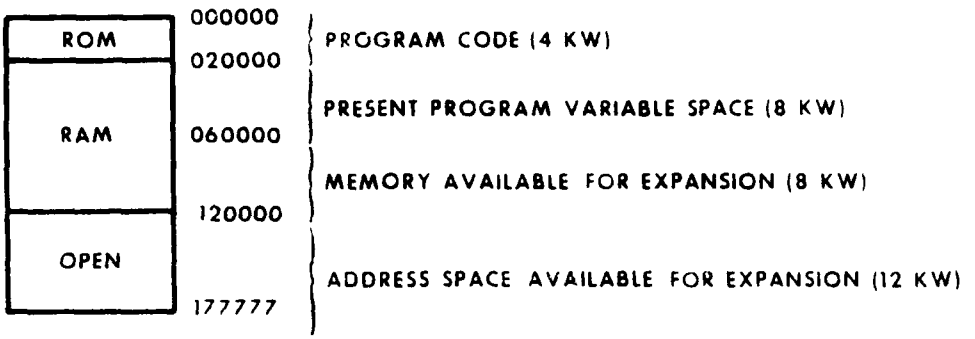


Fig. F1 - Voltmeter memory map.

The ROM code used by this version of the voltmeter occupies approximately 15300 (octal) bytes of memory, so there is room for expansion.

The program variable space allows, at present, data buffers of 1024 samples, but this can be expanded by modifying the parameter BSIZE in LSIVAR when the program is rebuilt. Note that for each sample a buffer is expanded, memory is used as follows:

6 bytes for the sampled A/D waveform (2 bytes per channel),
4 bytes for the integer to floating-point conversion buffer,
4 bytes for the floating-point data window.

This requires approximately 7K words of RAM memory for each additional 1024 samples.