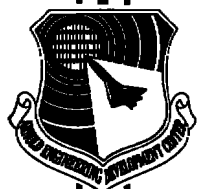


AEDC-TMR-91-V3

C.1



ANALYSIS SUPPORT FOR SPACE SYSTEMS AREA

S. L. Steely

Calspan Corporation/AEDC Operations

February 1991

PROPERTY OF U.S. AIR FORCE
AEDC TECHNICAL LIBRARY

TECHNICAL REPORTS
FILE COPY

Approved for public release; distribution is unlimited.



ARNOLD ENGINEERING DEVELOPMENT CENTER
ARNOLD AIR FORCE BASE, TENNESSEE
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE

Report Documentation Page				Form Approved OMB No. 0704-0188	
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE FEB 1991		2. REPORT TYPE		3. DATES COVERED 00-00-1991 to 00-00-1991	
4. TITLE AND SUBTITLE Analysis Support For Space Systems Area				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Calspan Corporation/AEDC Operations,Arnold Air Force Station,TN,37389				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT see report					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT Same as Report (SAR)	18. NUMBER OF PAGES 276	19a. NAME OF RESPONSIBLE PERSON
a. REPORT unclassified	b. ABSTRACT unclassified	c. THIS PAGE unclassified			

NOTICES

When U. S. Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise, or in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

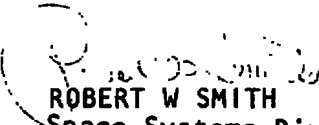
References to named commercial products in this report are not to be considered in any sense as an endorsement of the product by the United States Air Force or the Government.

DESTRUCTION NOTICE

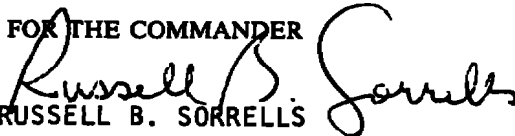
For classified documents, follow the procedures in DoD 5200.22-M, Industrial Security Manual, Section II-19 or DoD 5200.1-R, Information Security Program Regulation, Chapter IX. For unclassified, limited documents, destroy by any method that will prevent disclosure or reconstruction of the document.

APPROVAL STATEMENT

This report has been reviewed and approved.


ROBERT W SMITH
Space Systems Division
Directorate of Aerospace Flt Dyn Test
Deputy for Operations

Approved for publication:

FOR THE COMMANDER

RUSSELL B. SORRELLS
Analysis Manager
Directorate of Aerospace Flt Dyn Test
Deputy for Operations

SUMMARY

This report documents, in summary, project work conducted during FY89-90 to enhanced analysis support of Space Systems testing and evaluation. The project was originally prepared and written in response to the AEDC CN45PW Preliminary Engineering Authorization to support Focal Plane Array (FPA) testing in the Space Systems area. This project includes efforts in the Space Systems Area at AEDC for improving Focal Plane Array (FPA) testing and 12V optical signatures testing. The second and distinctly different work phase was added to the revised FY90 project plan to support the optical signatures analysis for 12V Space Chamber testing at AEDC. The project is currently scheduled to continue into FY92.

This project is a continuation of work which began during FY89 and terminates at the end of FY92. Several tasks were initiated during FY89 in order to develop a foundation for completing the project objectives. During FY89-90 the AEDC FPA experimental methodologies were reviewed and measurement theory principles investigated and compared with current FPA testing methodologies. Uncertainty reporting techniques have been thoroughly investigated to understand the nomenclature, its interpretation, and the uncertainty methodology. Experimental error sources were identified for investigation and characterization. An uncertainty spreadsheet was developed which can provide a systematic method of computing and reporting FPA-related measurement uncertainties. Elementary data- and function-plotting routines were developed, enhanced, and ported on the VAX systems and PC systems to provide a common graphical plotting environment and a methodology for supporting experimental data analysis and theoretical investigations. New versions of DISSPLA were installed on the PC and VAX systems to support general graphics and scientific visualization methods. New versions of SLATEC and VISAGE were also installed to support this effort. The SLATEC mathematical and scientific subroutines have also been ported to a PC for a common development environment using DISSPLA. The latest version (XV) of the Optical Signatures Code (OSC) was obtained and installed on a local VAX computer system for use at AEDC. Previous 12V signature tests have been reviewed for comparison with OSC predictions. A graphical user interface was developed to improve/provide user input to the OSC.

This report provides a summary of the project's Phase 1 work which supports the FPA effort. Several appendices have been prepared to segregate the topical elements for separate independent distribution. They are included here for completeness. As a result of the major computer network enhancements made to support the FPA effort, a "Space Systems Network User's Guide" was prepared and is included as an appendix. The second phase of the project is reported separately in the AEDC technical memorandum report AEDC-TMR-91-V1 [Selman 1991].

TABLE OF CONTENTS

SUMMARY	1
TABLE OF CONTENTS	2
LIST OF FIGURES	3
LIST OF TABLES	3
NOMENCLATURE	4
1 INTRODUCTION	5
2 FACILITY DESCRIPTION	6
2.1 STAND-ALONE FUNCTIONAL TESTING	6
2.2 SPECTRAL-RESPONSE TESTING	7
2.3 FLOODED-SOURCE TESTING	8
2.4 SPOT-SOURCE TESTING	10
2.5 DIRECT WRITE SCENE GENERATION TESTING	10
3 COMPUTATIONAL ENHANCEMENTS	11
3.1 SPACE SYSTEMS COMPUTATIONAL NETWORK	11
3.2 GRAPHICS VISUALIZATION AND NEW SOFTWARE	13
3.2.1 Optical Analysis and CODE V	13
3.2.2 DISSPLA	13
3.2.3 TPLOT	14
3.2.4 FUNPLOT/PLOTDATA	14
3.2.5 SLATEC	14
3.2.6 VISAGE	15
3.2.7 WordMARC	15
4 FPA UNCERTAINTY ANALYSIS	16
4.1 SCIENTIFIC UNCERTAINTY SPREADSHEET	18
4.1.1 Analytical Description	18
4.1.2 FPA Error Sources	20
5 CONCLUDING REMARKS	22
REFERENCES	24
Appendix A PC TPLOT	A-1
Appendix B VAX TPLOT	B-1
Appendix C PC PLOTDATA	C-1
Appendix D VAX PLOTDATA	D-1
Appendix E Student's "t" and Spreadsheet Example	E-1
Appendix F Space Systems Network User's Guide	F-1

LIST OF FIGURES

Figure 1. FPA cryogenic dewar assembly.

Figure 2. Spectral-response testing configuration.

Figure 3. Typical normalized spectral response.

Figure 4. Flooded-source testing configuration.

Figure 5. Typical responsivity plot.

Figure 6. Typical linearity plot.

Figure 7. Spot-source testing configuration.

Figure 8. DWSG FPA testing configuration.

Figure 9. Current Space Systems Ethernet configuration.

LIST OF TABLES

Table 1. Coefficients for the Student's "t" curve-fit.

NOMENCLATURE

$A(t,v)$	Student's probability function for a two-tailed distribution
B_r	Total bias limit (of a result)
B_i	Individual measurement bias limits
$B(a,b)$	Beta Function
$I_\gamma(a,b)$	Incomplete Beta Function
M	Radiant exitance
n	Sample size (number of measurements in a sample)
r	Result derived from some measurement(s)
$R(\lambda)$	Spectral responsivity
$R_{\max}(\lambda)$	Maximum spectral responsivity
S_r	Total precision index (of a result)
S_i	Individual measurement precision index
t_{95}	Student's two-tailed "t" value (95 percent level)
U_r	Uncertainty for a symmetrical uncertainty interval
U^-	Uncertainty corresponding to the lower value of the uncertainty interval
U^+	Uncertainty corresponding to the upper value of the uncertainty interval
x	Measured value
$\bar{x}$	Sample average of n measurements
θ_i	Influence coefficients
v	Degrees of freedom

1 INTRODUCTION

The work reported herein was sponsored by the Arnold Engineering Development Center (AEDC), Air Force Systems Command (AFSC), under Program Element Number 65807F, Control Number 9S01, at the request of the Directorate of Aerospace Flight Dynamics Test (DOF). The Air Force Project Manager was T. Nguyen (FY89-90). The work was performed by Calspan Corporation/AEDC Operations, operating contractor for the aerospace flight dynamics testing facilities at AEDC, AFSC, Arnold Air Force Base, Tennessee, under AEDC Project Number CN45VW. The project began in FY-89 and is currently scheduled to continue through FY92. This report summarizes Phase-1 work conducted during the FY89-90 period. The second phase of the project is reported separately in an AEDC technical memorandum report AEDC-TMR-91-V1 [Selman 1991].

Current efforts in the Space Systems Area at AEDC include improving Focal Plane Array (FPA) testing and 12V optical signatures testing. The FPA test facilities at AEDC are being upgraded to support and potentially support several programs: the Advanced Warning System (AWS), Brilliant Eyes, Brilliant Pebbles, Ground Surveillance and Tracking System (GSTS), Ground-Based Interceptor (GBI), Midcourse Space Experiment (MSX), Precursor Above The Horizon Sensor (PATHS), and the Hybrids With Advanced Yield For Surveillance (HYWAYS). The 12V Space Chamber has also recently undergone refurbishment and operational checkout, and a new cryogenic infrared circular variable filter spectrometer has been procured for upcoming tests. The 12V chamber is designed to support optical signature analysis for a variety of cryogenic environments. This project has specifically provided analytical and computational support for two specific areas. Phase 1 of the project plan is concerned with supporting the FPA test efforts at AEDC by providing new computational resources and investigating measurement uncertainty. Phase 2 of the project supports the 12V Aerospace Chamber optical signature testing and is discussed more completely in a recent AEDC TMR [Selman 1991].

Several aspects of the FPA testing effort have been addressed in this project's Phase-1 effort. Emphasis has been devoted to improving the FPA computational capabilities and specifically providing some analytical and graphical tools to enhance our capacity to perform FPA analysis, testing, and reporting with some emphasis also devoted to improving our FPA measurement uncertainty analysis capability. Several appendices are included that describe the plotting routines, an example uncertainty spreadsheet, and the Space Systems Network User's Guide.

Experimental measurements are fundamental to AEDC's successful FPA test facility. The credibility of reported results is contingent upon a thorough evaluation of experimental/testing objectives, experimental measurement requirements, associated measurement errors, and an error/uncertainty analysis. It is essential for AEDC

to provide a thorough characterization of experimental measurement errors and theoretical limitations involved in optical diagnostics, and to ultimately strive to minimize FPA measurement errors by using well-planned experimental designs and methods of statistical control. As a result of the volume of data generated during FPA tests, it is also necessary to establish statistical and graphical methods to render the data useful, readily comprehensible, and more easily understandable. New computational methods, graphical visualization, and statistical methods are therefore necessary to enhance our understanding of the experimental results and should be exploited to assist in revealing measurement errors, experimental trends, and correlated patterns in voluminous FPA data.

2 FACILITY DESCRIPTION

The Arnold Engineering Development Center's Space Systems testing effort has been expanded from just testing sensor systems to also testing Focal Plane Arrays. FPA testing at AEDC is performed using two distinctly different methods. One method uses standard broadband infrared "blackbody" sources and the other method employs non-traditional coherent laser sources. The standard broadband infrared blackbody FPA testing effort is composed of four types of primary testing configurations. FPAs generally undergo an initial and simple stand-alone evaluation to determine if they are functionally operational and have been properly installed. The FPAs are generally tested to determine their relative radiometric spectral response, flood-source tested to characterize other parameters such as broad-band radiometric responsivity, and spot-source tested to investigate individual pixel and pixel-to-pixel crosstalk response characteristics. A more detailed description can be found in other AEDC references [e.g. Whitehead 1988]. A new innovative FPA testing technology has also been recently developed at AEDC to provide advanced direct write scene generation (DWSG) using infrared laser sources that are attenuated but focused directly onto the FPAs, simulating complex background and target scenarios [Elrod 1989]. Each type of FPA testing has its own set of peculiar features that are addressed separately. The following subsections provide a brief summary of each type of FPA testing.

2.1 STAND-ALONE FUNCTIONAL TESTING

FPAs tested at AEDC are generally mounted in a Lakeshore Cryotronics, Inc. Modular Test Dewar for mechanical, electrical, and environmental support. The dewar provides a rigid mechanical support and radiation-baffled isolation to physically position and mechanically stabilize the FPA's location while being tested. The dewar assembly provides the necessary electrical connections to input bias and clocking signals for controlling the FPAs and to also extract the FPA's output signals. The dewar assembly also provides for two-stage cooling using liquid helium which is supplied from an externally connected helium dewar.

The FPA dewar without a vacuum cover is illustrated in Fig. 1. The dewar assembly is covered to provide a vacuum-tight enclosure to check the FPA's electrical connections, operational status, etc. A steady-state or electrically chopped light emitting diode (LED) provides a continuous or chopped infrared signal, respectively.

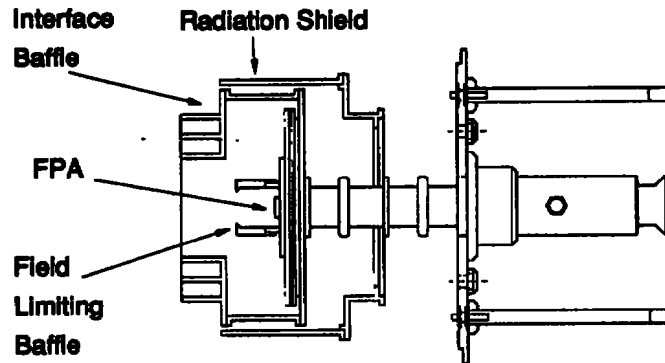


Figure 1. FPA cryogenic dewar assembly.

2.2 SPECTRAL-RESPONSE TESTING

The FPA dewar assembly is also used as an integral part of the spectral-response measurement station (Fig. 2). In this configuration, the dewar is covered with a vacuum-tight housing which contains an optical-grade, transmissive KRS-5 window and a number of cryogenically cooled apertures and baffles to suppress stray infrared radiation. A more detailed description of the hardware is available [Whitehead 1988].

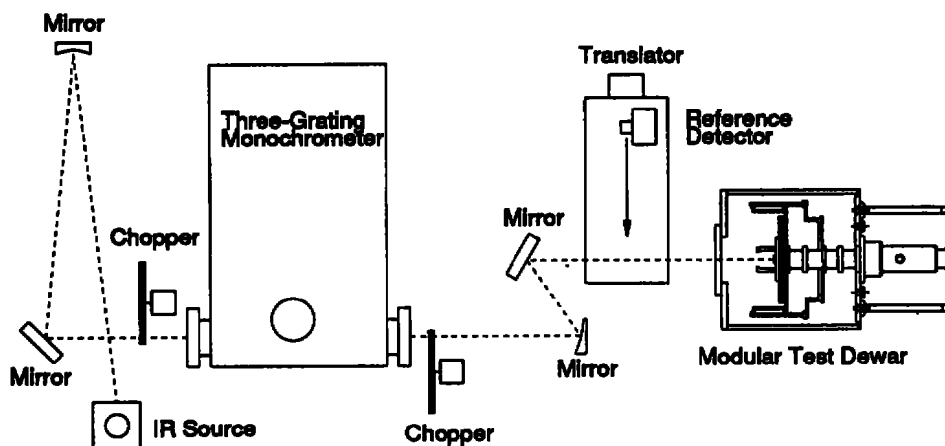


Figure 2. Spectral-response testing configuration.

The primary objective of the spectral response test configuration is to measure FPA's relative or normalized spectral responsivity $r(\lambda) = R(\lambda)/R_{\max}(\lambda)$, where $R(\lambda)$ is the responsivity and $R_{\max}(\lambda)$ is the maximum responsivity (see Fig. 3).

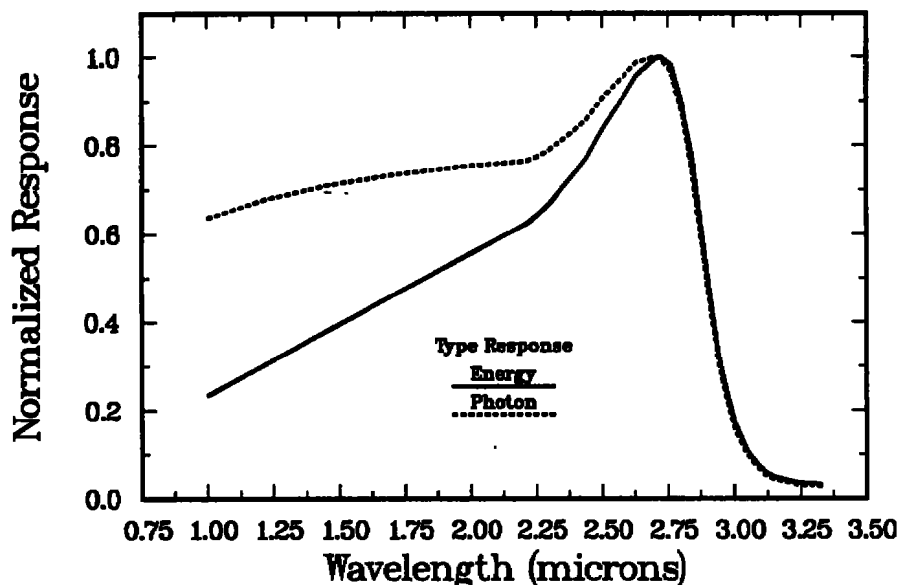


Figure 3. Typical normalized spectral response.

2.3 FLOODED-SOURCE TESTING

The flooded-source testing configuration is the primary mode of testing FPAs at AEDC. In this mode of test, the FPA remains mounted in the FPA dewar assembly but is, however, installed in a larger vacuum chamber as illustrated in Fig. 4. This

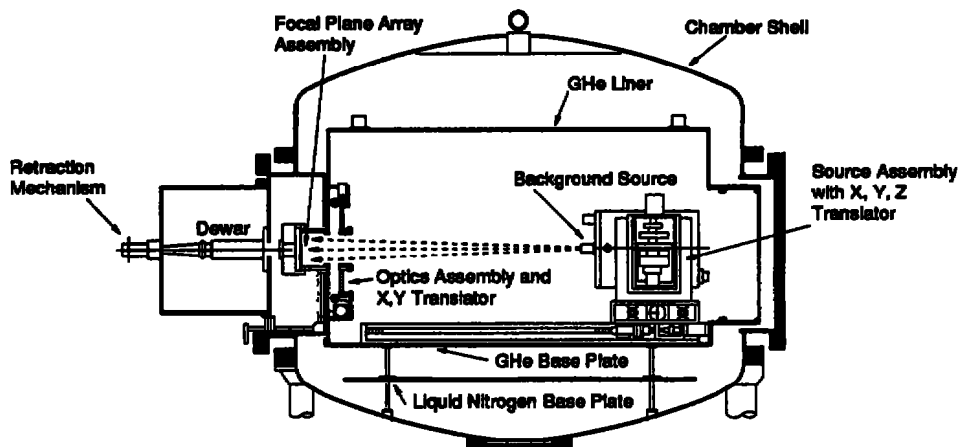


Figure 4. Flooded-source testing configuration.

chamber is referred to as the Focal Plane Characterization Chamber (FPCC). The FPCC supports the majority of FPA tests currently conducted at AEDC. The FPCC is vertically aligned and consists of right-circular, concentric cylinders. A large flange is attached to the side of the chamber to support the FPA dewar as illustrated in Fig 4. The chamber and dewar are evacuated and cryogenically cooled for FPA testing. The chamber contains a calibrated infrared source mounted on an X-Y-Z-axis translator for positioning and an optical assembly that is retracted out of the optical path during the flooded-source mode. In the flooded-source mode, radiation emitted by the infrared source package "uniformly" irradiates the FPA being tested. The source package contains the infrared source which can be changed in temperature to modify the radiant exitance, M , and a number of source aperture sizes can be selected. The source package can also be positioned at a variable distance from the FPA by a Z-axis translation. By selectively varying the source temperature, aperture size, and the source-to-FPA distance, the "uniform" irradiance E on the FPA can be varied for controlled FPA characterization. A more detailed description of this procedure can be found in Whitehead 1988 and Nicholson 1990.

In the flooded-source testing mode a number of FPA parameters (noise equivalent input, responsivity, dynamic range, linearity, radiometric stability, etc.) are evaluated. The results are often presented graphically to display the functional dependance on testing variables such as bias voltage, etc. (as illustrated in Fig. 5 and 6). Most data packages provided to the user contain tabular data plus numerous graphical plots that show trends and patterns in the characteristics and parameters of the FPAs. In addition to the AEDC references, a number of other good technical references are also available that describe detectors, FPA parameters, and related measurement principles [Dereniak 1984, Grum 1979, Keyes 1980, and Kingston 1978].

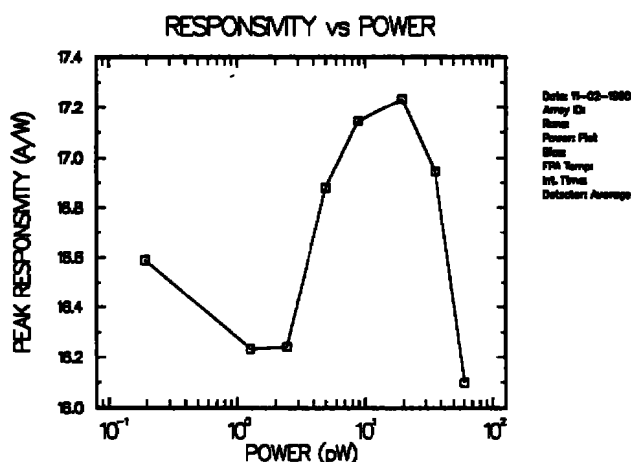


Figure 5. Typical responsivity plot.

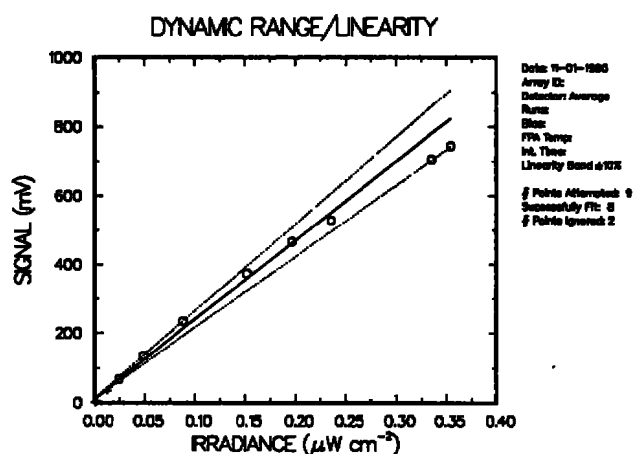


Figure 6. Typical linearity plot.

2.4 SPOT-SOURCE TESTING

The spot-source testing configuration is also performed in the FPCC but the optical assembly is injected into the optical path between the infrared source assembly and the FPA as illustrated in Fig. 7.

In the spot-source testing mode, the infrared source is imaged onto the FPA itself. The source image provides "individualized" pixel irradiation to primarily assess pixel-to-pixel crosstalk properties. In this mode of testing the source aperture radiant exitance, M , is attenuated by filter-wheel inserting an optical diffuser and spectral filter. Only a few of the total pixels are normally evaluated.

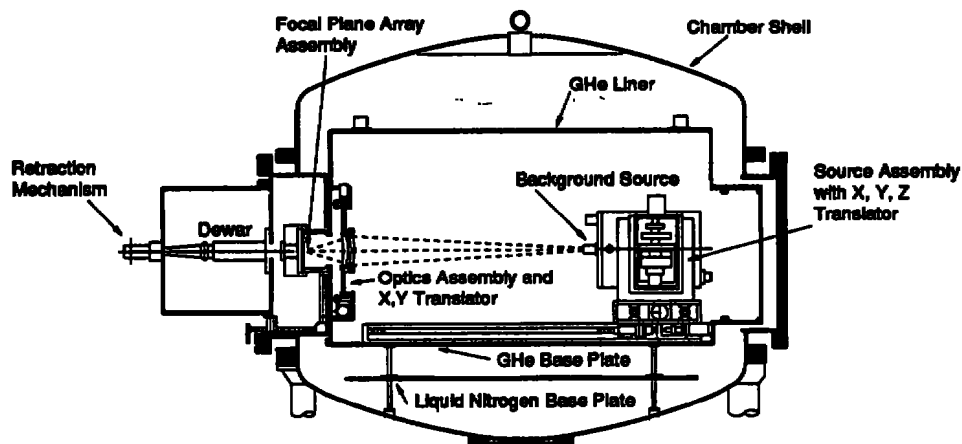


Figure 7. Spot-source testing configuration.

2.5 DIRECT WRITE SCENE GENERATION TESTING

Direct Write Scene Generation (DWSG) is a new technology used at AEDC that provides advanced mission simulation capabilities to evaluate many of the more mission-related performance parameters such as background suppression, clutter rejection, target discrimination, multi-target tracking, etc. The DWSG methodology, in contrast to the "blackbody" methods described previously, uses infrared laser sources that are directly focused onto the FPA. The DWSG technology has been chosen as a preferred method to reduce space sensor development risks by developing the capability to test infrared sensor focal plane arrays and data subsystems using "realistic" mission scenarios. The DWSG testing is in addition to and complimentary to the traditional FPA characterizations method which uses "blackbody" sources installed in the FPCC described earlier.

The DWSG method of testing also uses the FPA dewar assembly in a stand-alone configuration as illustrated in Fig. 8. The method employs an infrared laser; the laser beam is electro-optically and acousto-optically modulated to temporally and spatially modulate (x-y scanned) the infrared laser beam before it is subsequently focused onto the FPA using a set of scan optics.

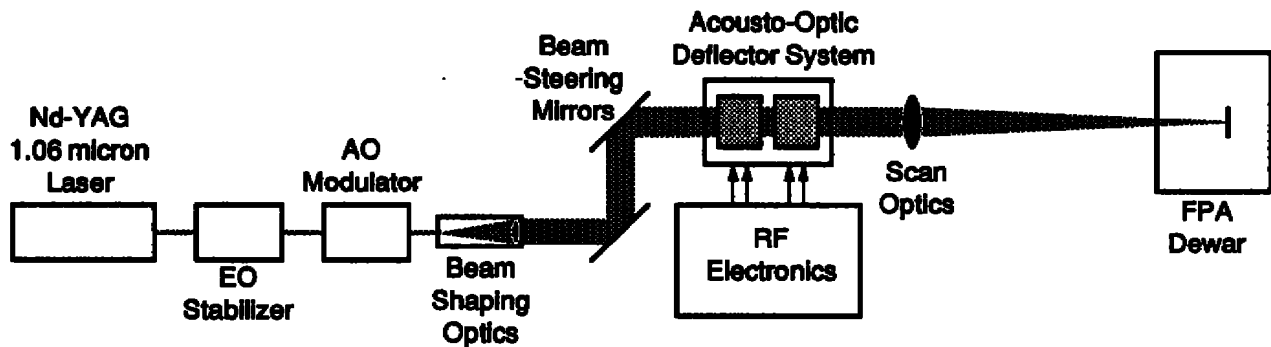


Figure 8. DWSG FPA testing configuration.

3 COMPUTATIONAL ENHANCEMENTS

3.1 SPACE SYSTEMS COMPUTATIONAL NETWORK

The Space Systems Network is primarily an Ethernet computer network of computer systems and currently consists of several MicroVAX and VAXstation computer systems interconnected by thick- and thin-wire Ethernet hardware and software. The network hardware and interconnected computer systems are located at Arnold Air Force Base in the Engineering Laboratory Addition (ELA). The network system serves a variety of scientific functions in support of Space Systems analysis, technology, and testing activities. This includes many forms of data acquisition/reduction, scientific analysis, and technical research which can be rendered solvable using VAX/VMS class of computers.

Many hardware and software additions/changes have been made to the computer network to support Focal Plane Array testing. Some locally developed graphics software source is contained in Appendix A-E; a Space Systems Network User's Guide has been written to describe most of the new features and is contained in Appendix F. The user's guide has specifically been written in a manner to facilitate future updates to individual sections. The dynamic evolution of the Space Systems Ethernet network hardware and software configuration makes it difficult to maintain current user information. Updates to the user's guide will be provided as necessary. The Space Systems network configuration was reconfigured to support a new FPA

3.2 GRAPHICS VISUALIZATION AND NEW SOFTWARE

Several new software packages have been developed and/or acquired in this project to support the Space Systems testing efforts with emphasis on the FPA testing needs. The software is for optical analysis, statistical analysis, and graphical display and documentation.

3.2.1 Optical Analysis and CODE V

One of the essential, original, and primary needs identified was optical design and analysis. Past efforts in Space Systems testing have required considerable optical analysis for facility design, technology, and testing programs. AEDC optical design and analysis requirements extend from the ultraviolet to the infrared region of the electromagnetic spectrum. Past and current AEDC design efforts also include refractive and reflective optical systems. Special emphasis has been devoted to design and analysis of high-quality diffraction-limited optical elements that require extensive computational optimization and diffraction analysis capabilities. Current requirements include design and analysis of large off-axis, wide-field-of-view reflective optical systems; infrared reflective collimators; visible and infrared laser-beam alignment systems; infrared scene generator reflective optical projector elements; and wide-field-of-view refractive achromatic optical imagers.

Current and typical AEDC efforts supported include the Aero-Optics Facility Development, Focal Plane Array testing, Space Systems Facility optical design, HEDI testing, and the LWIR Environment Threat Simulator program. Efforts were devoted in this project to provide and maintain state-of-the-art optical design and analysis capabilities to support the ever increasing optical testing demands. The latest version (7.31) of the optical analysis and design software, CODEV, has been acquired and installed on a VAXstation III system to support AEDC analysis and testing needs. User information for the CODEV program is provided in the Space Systems Network User's Guide.

3.2.2 DISSPLA

The DISSPLA graphics libraries were acquired to support scientific programming that required simple and advanced graphics output using systems that range from PCs to CRAYs at AEDC. DISSPLA is a high-level FORTRAN graphics subroutine library for interactive or batch data representation applications. DISSPLA's integrated subroutines allow programmers to generate graphs, charts, maps, surfaces, contours, and 2- and 3-dimensional designs. DISSPLA's complete functionality, flexibility, and productivity have made it an excellent choice for a wide range of applications. Options currently licensed for use at AEDC include Shaded Fonts, Page Layout, Business Features, Dynamics, Mapping, Post-Processor, Contouring, and GKS.

Previous versions of DISSPLA were installed on the MicroVAX II system. The software license is currently maintained through an AEDC sight license; DISSPLA 11.0 was installed on all Space Systems VAX computers for general use and to support the FPA graphical display and documentation requirements. DISSPLA was originally chosen to maintain compatibility with previous programs used on other AEDC scientific computer systems (CRAYs, VAXes, etc.). The Space Systems network-peculiar user information for accessing the DISSPLA subroutines is provided in the Space Systems Network User's Guide.

3.2.3 TPLOT

A generalized data graphics plotting subroutine has been developed for use on the Space Systems Ethernet VAX computers and PC systems for users that do not desire or have time to learn the DISSPLA command language. The two versions of TPLOT (PC and VAX) provide the same functionality. The VAX version uses DISSPLA 11.0 and the VAX FORTRAN compiler. The PC version, however, uses DISSPLA 10.5 and requires a Ryan-McFarland FORTRAN compiler. TPLOT's general use has been extended to support the graphics display and output for the FPA user interface. Specific TPLOT user information has been included in the Space Systems Network User's Guide. The current PC and VAX TPLOT software subroutines are provided and included in Appendices A and B, respectively.

3.2.4 FUNPLOT/PLOTDATA

Two other useful PC and VAX programs have also been developed to support function and data plotting and scientific research related to the FPA effort at AEDC. FUNPLOT and PLOTDATA were developed and use the general features of the TPLOT graphics subroutine to easily plot functions and data from user input files. The function and data plots included in this report were, as examples of the type of output generated, prepared using these new graphical resources. Appendices C and D, respectively, contain the PC and VAX versions of the PLOTDATA programs in their current forms.

3.2.5 SLATEC

SLATEC is a large collection of FORTRAN mathematical subroutines brought together in a joint effort by the Air Force Weapons Laboratory, Lawrence Livermore National Laboratory, Los Alamos National Laboratory, Magnetic Fusion Energy Computing Center, National Institute of Standards and Technology, Sandia National Laboratories (Albuquerque and Livermore), and the Martin Marietta Energy Systems Incorporated at Oak Ridge National Laboratory.

SLATEC is characterized by portability, good numerical technology, good documentation, robustness, and quality assurance. The Library is divided into the following subsections following the guidelines of the NBS (now NIST) classification system: Elementary and Special Functions, Elementary Vector Operations, Solutions of Systems of Linear Equations, Eigenanalysis, QR Decomposition, Singular Value Decomposition, Interpolation, Solution of Nonlinear Equations, Optimization, Quadrature, Ordinary Differential Equations, Partial Differential Equations, Fast Fourier Transforms, Approximations, Psuedo-Random Number Generation, Sorting, Machine Constants, and Diagnostics and Error Handling.

SLATEC, version 3.2, was acquired from the Air Force Weapons Laboratory for use at AEDC on the CRAY, PC, and VAX computer systems. **SLATEC** 3.2 is available and used on the Space Systems VAX computers and PCs. PC-based link libraries have been developed for the Microsoft and Ryan-Macfarland FORTRAN compilers. **SLATEC** user information and **SLATEC**-Help support programs installed for use at AEDC are described in the Space Systems Network User's Guide.

3.2.6 VISAGE

The VAX Interactive Signal Analysis and Graphics Environment (**VISAGE**) program was developed jointly by Applied Technology Associates (ATA) and the Air Force Weapons Laboratory (AFWL).

VISAGE is a single program developed on and for a VAX/VMS system using the VAX-FORTRAN compiler. **VISAGE**'s capabilities range from the simple plotting of time history data, to computing and plotting power spectral densities, to computing coherence and cross spectral densities, as well as many other signal processing operations. **VISAGE** has very extensive graphing capabilities. The current version (4.3) has been acquired and modified to support the DISSPLA 11.0 graphics post-processor and the generation of device independent output (metafiles).

VISAGE 4.3 was acquired from AFWL for use at AEDC on the Space Systems Ethernet network VAX computer systems. The 4.3 version of **VISAGE** is currently installed, and its use is explained in the Space Systems Network User's Guide.

3.2.7 WordMARC

General scientific documentation and presentation requirements are partly supported using the **COMPOSER+** software. **COMPOSER+** is the latest version of the scientific wordprocessor produced by **MARC** software. **COMPOSER+** supports mathematical copy and graphics composition. It is especially useful for scientific reporting and general documentation preparation. Current use includes excellent generation of presentation quality viewgraphs.

A new version of COMPOSER+ is designed to run on many systems (including PCs) and was installed and maintained for general use on the Space Systems Ethernet MicroVAX II (node MICROS) computer. No future upgrades to this software are scheduled as a result of the increased availability of scientific documentation software for PC systems. A full description of its operation and the local options added are contained in the Space Systems Network User's Guide.

4 FPA UNCERTAINTY ANALYSIS

Interest in experimental design, diagnostic accuracy, and error analysis has increased as a result of the inherent complexity of radiometric measurements, electro-optical sensor tests, and other optical diagnostic technologies used at AEDC. Optical and FPA diagnostics and testing require a thorough characterization of experimental measurement errors. Experimental measurement errors are typically categorized as either bias-type (systematic) or precision-type (random) measurement errors. The purpose of an uncertainty analysis/statement is to estimate the interval within which the "true" value of the result being reported is expected to be contained.

If we report a measurement, x , of some "assumed constant property", a sample average, $\bar{x}$, (in a simple case the sample may consists of n measurements taken from some assumed or specified larger population), or some result, r , derived from measurements, then the uncertainty interval expected to contain the "true" value is sometimes denoted by

$$\bar{x} \pm U \quad (\text{sometimes denoted as } \bar{x} \pm U_{\bar{x}}) \quad (1)$$

where the average value, as an example for illustration, may be considered important and chosen for reporting [ANSI/ASME 1986, ISO 1987, and Coleman 1989]. To say that we expect the uncertainty interval to contain the "true" value means that if many uncertainty intervals were constructed from similar conditions and measurements then a certain fraction (e.g., 0.95 for the current URSS model) of the uncertainty intervals are expected to contain the "true" value. The particular uncertainty interval reported may or may not necessarily contain the "true" value [Wilson 1952 and Coleman 1989].

By standards, U is denoted as the uncertainty of the value being reported, in this example $\bar{x}$. A point to note is that it is convenient to chose U to be non-negative in order that $\bar{x} - U \leq \bar{x} + U$ and that the desired interval is consistently interpreted, represented, and properly used in any type of uncertainty statement, more importantly in Monte Carlo measurement simulations where proper signs and interval tests are critical [Abernethy 1985, Taylor 1982, ANSI/ASME 1986, ISO 1987]. If the uncertainty interval is not symmetrical then the lowest value of the uncertainty interval is

denoted by $\bar{x} - U^-$ and the upper value of the uncertainty interval is denoted by $\bar{x} + U^+$.

4.1 UNCERTAINTY SPREADSHEET

This project has initiated an effort to help better understand the FPA testing results and to help provide some understanding of the inherent FPA measurement errors and how to determine and report the uncertainty of the reported results. Each testing effort is different, and different FPA configurations and testing requirements change from test to test, requiring different and new uncertainty analyses for each test. There are, however, some common factors in the FPA tests. To exploit these common factors a method has been developed to determine and present the uncertainty in a systematic manner using the formatting and systematic calculations of an uncertainty spreadsheet. The uncertainty spreadsheet has been developed to allow a quick update of the calculations for items that may happen to change for a particular test. This method will maintain a log of the current measurement uncertainties, can be easily updated if the measurement method changes, and will provide a report-ready output for data packages, TMRs, TSRs, or TRs as needed.

4.1.1 Analytical Description

The uncertainty spreadsheet was developed using the standard ANSI and ISO uncertainty reporting methodology, Eq (1). Both of the uncertainty models [ANSI/ASME 1986] are included in the spreadsheet and are computed for some result, r , as

$$U_{rADD} = B_r + \frac{t_{95} S_r}{\sqrt{n}} \quad (2)$$

$$U_{rRSS} = \left[B_r^2 + \frac{(t_{95} S_r)^2}{n} \right]^{1/2} \quad (3)$$

where B_r is the bias limit of the result, S_r is the precision index of the result, t_{95} is the two-tailed Student's "t" value, and n is the sample size [ANSI/ASME 1986].

The symmetrical bias limit B_r and the precision index S_r are determined using the individual measurement bias limits B_i and the precision indices S_i

$$B_r = \left(\sum_i (\theta_i B_i)^2 \right)^{1/2} \quad (4)$$

$$S_r = \left(\sum_i (\theta_i S_i)^2 \right)^{1/2} \quad (5)$$

where θ_i is the influence coefficient defined as $\theta_i = \partial r / \partial P_i$, each B_i and S_i correspond to a parameter P_i , and $r = r(P_1, P_2, P_3, \dots)$ is a result being reported.

The degrees of freedom, ν_r , used in calculating the two-tailed Student's "t" value t_{95} , is determined using the Welch-Satterthwaite approximation formula

$$\nu_r = \frac{\left[\sum_i (\theta_i S_i)^2 \right]^2}{\sum_i \frac{(\theta_i S_i)^4}{\nu_i}} \quad (6)$$

The Student's "t" values, t_{95} , are automatically calculated by the spreadsheet using a curve-fit to the t_{95} values obtained using an iterative solution with the Incomplete Beta Function form for the t_{95} values. The mathematical form can be written as

$$A(t, \nu) = \frac{1}{\nu^{1/2} B(1/2, \nu/2)} \int_{-t}^t \left(1 + \frac{x^2}{\nu} \right)^{-(\nu+1)/2} dx \quad (7)$$

where $B(1/2, \nu/2)$ is the Beta function [Kendall 1969]. Equation (7) can be written in a different form using the Incomplete Beta function $I_\gamma(a, b)$.

$$A(t, \nu) = 1 - I_\gamma(\nu/2, 1/2), \quad \left(\text{where } \gamma = \frac{\nu}{\nu + t^2} \right) \quad (8)$$

Setting A equal to 0.95 for each degree of freedom, ν , Eq. (8) was iteratively solved to obtain the desired t_{95} values. The t_{95} values are computed in the uncertainty spreadsheet using Eq. (9), a least-squares curve-fit approximation of the values from Eq. (8). The coefficients a_i for Eq. (9) are given in Table 1.

$$t_{95} = a_0 + a_1 \nu^{-1} + a_2 \nu^{-2} + a_3 \nu^{-3} + a_4 \nu^{-4} + a_5 \nu^{-5} \quad (9)$$

The least-squares curve-fit used in the spreadsheet is a simpler but rather accurate equation to obtain t_{95} for given values of ν in the spreadsheet. Table 1

illustrates the curve-fit results and accuracy. The program used to obtain these results and an example of the spreadsheet's output format are provided in Appendix E.

Table 1. Coefficients for the Student's "t" curve-fit.

$a_0 = 1.959963985+00$ (specified) $a_1 = 2.37369801D+00$ $a_2 = 2.78200513D+00$ $a_3 = 2.89980933D+00$ $a_4 = 4.40998341D-01$ $a_5 = 2.24972976D+00$		
V	FIT, Eq. (9)	t95, Eq. (8)
1.00000E+00	1.27062E+01	1.27062E+01
2.00000E+00	4.30266E+00	4.30265E+00
3.00000E+00	3.18241E+00	3.18245E+00
4.00000E+00	2.77649E+00	2.77645E+00
5.00000E+00	2.57061E+00	2.57058E+00
6.00000E+00	2.44691E+00	2.44691E+00
7.00000E+00	2.36461E+00	2.36462E+00
8.00000E+00	2.30599E+00	2.30600E+00
9.00000E+00	2.26214E+00	2.26216E+00
1.00000E+01	2.22812E+00	2.22814E+00
1.10000E+01	2.20097E+00	2.20099E+00
1.20000E+01	2.17880E+00	2.17881E+00
1.30000E+01	2.16036E+00	2.16037E+00
1.40000E+01	2.14478E+00	2.14479E+00
1.50000E+01	2.13145E+00	2.13145E+00
1.60000E+01	2.11990E+00	2.11991E+00
1.70000E+01	2.10982E+00	2.10982E+00
1.80000E+01	2.10093E+00	2.10092E+00
1.90000E+01	2.09303E+00	2.09302E+00
2.00000E+01	2.08597E+00	2.08596E+00
2.20000E+01	2.07388E+00	2.07387E+00
2.40000E+01	2.06391E+00	2.06390E+00
2.60000E+01	2.05554E+00	2.05553E+00
2.80000E+01	2.04842E+00	2.04841E+00
3.00000E+01	2.04229E+00	2.04227E+00
4.00000E+01	2.02109E+00	2.02108E+00
5.00000E+01	2.00857E+00	2.00856E+00
6.00000E+01	2.00031E+00	2.00030E+00
7.00000E+01	1.99445E+00	1.99444E+00
8.00000E+01	1.99008E+00	1.99006E+00
9.00000E+01	1.98669E+00	1.98667E+00
1.00000E+02	1.98398E+00	1.98397E+00
2.00000E+02	1.97190E+00	1.97190E+00
1.00000E+03	1.96234E+00	1.96234E+00

4.1.2 FPA Error Sources

Some FPA-testing error sources have been identified for investigation. Other error sources will be considered as they are identified for particular test situations. Error sources vary from test to test and depend on particular operational aspects of

the testing environment, the particular type of detector/FPA being tested, and the test objectives (single detector, detector array, monolithic vs. hybrid array; photoconductive vs. photovoltaic mode; correlated vs. uncorrelated processing; noise equivalent input, responsivity, dynamic range/linearity, radiometric stability, power consumption; etc.). The error sources included and described in the project plan range from those in the NIST calibration transfer of AEDC's SIRS II reference blackbody to those inherent in quantum-mechanical electromagnetic fluctuations that introduce photon/shot noise in the FPA detection process itself.

NIST SIRS2 Blackbody Calibration: The infrared blackbody sources used at AEDC are traceable to the National Institute of Standards and Technology (NIST). An AEDC standard infrared source (SIRS) was originally designed (Sherrell 1981) for transferring a radiometric calibration from NIST. FPA characterization testing at AEDC, that requires measurement traceability, is commonly accomplished using a new standard infrared source (SIRS2) that has also been calibrated using a working standard blackbody source that is calibrated at NIST; this provides direct traceability to NIST. The NIST low-background infrared (LBIR) calibration facility and its absolute cryogenic radiometer (ACR) are used in the calibration process [Ebner 1989]. The design and description of the blackbody source is available from Sherrell [1981] and Jarrett [1989].

Specific data related to calibration of the SIRS2 blackbody is contained in the NIST calibration reports provided to AEDC. The primary error sources previously considered by NIST only included those resulting from geometrical/metrology errors and diffraction-correction errors. The metrology-type error sources NIST used in the NIST uncertainty reporting included the separation distances between apertures and the aperture diameters. AEDC and NIST scientists have discussed additional error sources (photon, instrumentation, etc.) for NIST to include in future blackbody calibration uncertainty analyses. Discussions with NIST scientists include use of the diffraction-correction procedure, which is currently used by NIST and AEDC. Inappropriate application of the diffraction-correction model can result in a bias error of considerable relative magnitude (several percent). The standard diffraction models used by NIST and AEDC are derived from the Fresnel-Kirchhoff diffraction formula with the obliquity factor assumed approximately equal to one; this assumption implies that the apertures and blackbody sources are all small in angular extent, which is contrary to many AEDC IR source designs.

AEDC Bolometric Blackbody Calibration Transfer: The NIST-calibrated SIRS2 blackbody source is used as a working standard for FPA testing at AEDC. The SIRS2 is used to transfer the calibration to a particular blackbody used in testing. The blackbody calibration transfer is a null-calibration process that compares the blackbody being calibrated with the SIRS2 working standard. A bolometer is used for the null comparison test [Lienemann 1984, Herron 1990, and Jarratt 1990]. A number of error sources are associated with the bolometric transfer process: blackbody source

photon fluctuations (Boson factor included for large temperatures), source aperture diameters, source aperture cryogenic thermal contractions, source aperture cryo-deposition, source aperture thermal gradients, blackbody temperatures, blackbody source-to-pixel distance, blackbody point-source assumption, blackbody's aperture normal-to-pixel angle, blackbody's spectral transmission losses, temperature phonon noise (bolometer related), electronic noise from internal electromagnetic interference, stray radiation, chopper stability, instrumental errors, and diffraction-correction factor calculation.

NIST Spectral Calibration: There is currently no good direct calibration path for the relative spectral response measurements made for the FPA. This matter is being discussed with NIST personnel and has been a topic presented during previous LBIR working group meeting.

Instrumentation Error Sources: The FPA testing effort includes the use of a number of measurement instruments. These instruments are generally calibrated at AEDC. A notebook has been prepared that is to be updated to contain current instrumentation calibration information. Each measurement instrument used in the FPA measurement process has an accuracy and a range of operation associated with it. This information is to be included in the total uncertainty analysis effort. A number of measurement instruments are used in the FPA testing procedure:

- Fluke 8502A digital voltmeter (dewar temperature, focal plane voltage, temperature sensors, current sources, house keeping chamber temperature, position indicators, and platinum temperature sensors for the blackbody temperature)
- Constant current source
- Preamp-A/D with noise board (20 channel system)
- Keithley 617 Electrometer (focal plane currents)
- HP 3325 synthesizer (timer clock)
- HP 3582 spectrum analyzer (for detector arrays and noise measurements)
- MDF noise generator (bandwidth measurements, etc.)
- Aerotec (positioners)
- Boonton 93AD rms meter

Blackbody Source Related Errors During Testing: The blackbodies used for testing are associated with a number of error sources similar to those encountered in the NIST SIRS2 calibration and the AEDC null calibration process. A number of error sources have been identified for further investigation: source photon fluctuations (normal and Boson factors for large temperatures), source aperture diameters, cryo-deposition source area degradation resulting from extended testing periods, source aperture cryogenic/thermal contractions, source aperture thermal gradients, blackbody temperature, blackbody source-to-pixel distance, blackbody point-source assumption, blackbody aperture normal-to-pixel angle, and blackbody spectral transmission losses.

FPA Geometry Related: There are a number of error sources that are related to the geometry of the FPA itself and include FPA pixel normal-to-source angle, Off-axis position of pixels relative to the optical centerline, and the pixel area.

Detector-Generated Fluctuations: The detectors have inherent noise sources that will contribute to the total measurement-to-measurement fluctuations and include Johnson (all detectors, resistive components, load resistors, etc.), 1/f noise, generation (photovoltaic and photoconductive detectors), recombination (photoconductive detectors only), and microphonic.

FPA Charge-Transfer-Related Noise: The focal plane arrays have noise sources that also contribute to the total FPA measurement-to-measurement fluctuations. The charge transfer devices used in the hybrid arrays introduce a number of error sources: input noise, transfer inefficiency noise, trapping noise, dark-current noise, clock feedthrough noise, floating diffusion reset noise, amplifier noise, detector uniformity noise, and read noise.

Chamber Related Noise: The FPACC introduces a number of error sources that contribute to the total background noise level. These noise sources add to the total NEI during the noise measurements. Those to be considered are electromagnetic radiation (EMI). FPACC stray background radiation photon noise (hot sources and 20 K background sources), and microphonic.

Data-Reduction Related Errors: The data acquisition and reduction portion of the measurement process includes a number of error sources that contribute to the total uncertainty of the final results and include A/D static quantization errors, A/D dynamic timing jitter (aperture uncertainty), A/D dynamic differential nonlinearity, A/D dynamic integral nonlinearity, data reduction roundoff, data reduction truncation, and numerical curve-fit errors.

5 CONCLUDING REMARKS

Uncertainty reporting techniques have been investigated to understand the nomenclature, its interpretation, and the uncertainty methodology. Experimental error sources were identified for investigation and characterization. An uncertainty spreadsheet was developed which will provide a systematic method of computing and reporting FPA-related measurement uncertainties. Elementary data- and function-plotting routines were developed, enhanced, and ported on the VAX systems and PC systems to provide a common graphical plotting environment and a methodology for supporting experimental data analysis and theoretical investigations. New versions of DISSPLA were installed on the PC and VAX systems to support general graphics and scientific visualization methods. New versions of SLATEC and VISAGE were also

installed to support this effort. The SLATEC mathematical and scientific subroutines have been ported to a PC for a common development environment using DISSPLA. As a result of the major computer network enhancements made to support the FPA effort, a "Space Systems Network User's Guide" was prepared and is included as an appendix.

The latest version (XV) of the Optical Signatures Code (OSC) was obtained and installed on a local VAX computer system for use at AEDC. Previous 12V signature tests have been reviewed for comparison with OSC predictions. A graphical user interface was developed to improve/provide user input to the OSC. This Phase-2 project effort was reported separately.

This project is to continue through FY92 and is to complete the error source investigation, the error/uncertainty analysis, and provide support for the 12V optical signatures testing effort.

REFERENCES

- Abernethy, R. B. and B. Ringhiser. "The History and Statistical Development of the New ASME-SAE-AIAA-ISO Measurement Uncertainty Methodology." AIAA-85-1403, July 1985.
- ANSI/ASME PTC 19.1, "Measurement Uncertainty." Performance Test codes Supplement, 1986.
- Apostol, T. M. *Mathematical Analysis*. Addison-Wesley Publishing Company, Reading, Massachusetts, January 1975.
- Campbell, N. R. *Physics the Elements*. Cambridge University Press, 1920.
- Campbell, N. R. *Foundations of Science The Philosophy of Theory and Experiment*. Dover Publication, New York, 1957.
- Coleman, H. W. and W. G. Steele. *Experimentation and Uncertainty Analysis for Engineers*. John Wiley & Sons, New York, 1989.
- Courant, R. *Differential and Integral Calculus*. Vol. I and II, Interscience Publishers, New York, 1937.
- Courant, R. and H. Robbins. *What is Mathematics*. Oxford University Press, New York, 1969.
- Dereniak, E. L. and D. G. Crowe. *Optical Radiation Detectors*. John Wiley, New York, 1984.
- Doebelin, E. O. *Measurement Systems*. McGraw-Hill, New York, 1975.
- Ebner, S. C. and A. C. Parr. "Update on the Low Background IR Calibration Facility at the National Institute of Standards and Technology." SPIE Vol. 1110, 1989.
- Elrod P. D., H. S. Lowry, C. B. Herron, and R. J. Johnson. "Development of a Direct Write Scene Generator." AEDC-TMR-89-V17, September 1989.
- Fulks, W. *Advanced Calculus*. John Wiley & Sons, New York, 1969.
- Grum, F. and R. J. Becherer. *Optical Radiation Measurements*. Volume 1, Academic Press, New York, 1979.

REFERENCES (CONTINUED)

- ISO TC30 SC9, "Fluid Flow Measurement Uncertainty." (Revision of ISO/DIS 5168), Draft copy as of May 1987.
- Jarratt, K. B., R. P. Young, and C. B. Herron "Development of a Cryogenic and Vacuum Compatible 1200 K Blackbody." AEDC-TR-90-13, August 1990.
- Keyes, R. J. *Optical and Infrared Detectors*. Springer-Verlag, New York, 1980.
- Kendall, G. K. and A. Stuart. *The Advanced Theory of Statistics*. (three volume set). Hafner Publishing Company, New York, 1969.
- Kingston, R. H. *Detection of Optical and Infrared Radiation*. Springer-Verlag, New York, 1978.
- Lienemann, K. A. "Calibration of a Germanium Bolometer for use in Cold-Background Radiometric Testing." AEDC-TMR-84-V26, September 1984
- Nicholson, R. A., C. L. Steele and M. A. Feder. "AEDC Focal Plane Array Test Capability and HYWAYS Test Methodology." AEDC-TR-90-5, April 1990.
- Selman, J. D. "AEDC Installation and Software Support of the Optical Signatures Code (OSC)." AEDC-TMR-91-V1, January 1991.
- Sherrell, F. G. "Infrared Standards to Improve Chamber TV Beam Irradiance Calibrations." AEDC-TR-80-45, January 1981.
- Taylor, John R. *An Introduction to Error Analysis*. University Science Books, Mill Valley, California, 1982.
- Whitehead, G. L., P. D. Elrod, K. B. Jarratt, and R. P. Young. "Development of a Focal Plane Array Test Capability." AEDC-TR-87-41, May 1988.
- Wilson, E. B. *An Introduction to Scientific Research*. McGraw-Hill, New York, 1952.

Appendix A

PC Version of TPLOT

```
*****
*           PC Version of TPLOT Source Code as of January 1991           *
*                                                                           *
*   MC = max curves   MS = max story lines   MM = max messages           *
*   IPL = legend packing array size   IPS = story packing array size     *
*****
```

```
SUBROUTINE TPLOT(NCURV,X,Y,NP)
PARAMETER (MC=50, MS=20, MM=10, IPL=MC*20, IPS=MS*20)
CHARACTER LET*2, ALINE*80
LOGICAL POWER10, ERRORS, MODIFIED
LOGICAL LTITLE(60), LXLBL(60), LYLBL(60), LSTRTMP(60), LLEGTIT(20)
CHARACTER*60 STRTMP, TITLE, XLBL, YLBL, MESTXT(MC)
CHARACTER*59 LEGTIT*20, LEGTXT(MC), STYTXT(MS)
COMMON/STORY/NSTYLS, SIZSTY, STYTXT, XSTYF, YSTYF
COMMON/LEGEND/XLEGF, YLEGF, SIZLEG, LEGTIT, LEGTXT
COMMON/LINES/LBOLD(MC), LCOLOR(MC), LTYPE(MC),
:                                LSYMBOL(MC), LSKIP(MC), SIZSYM
COMMON/LABEL/XTITF, YTITF, SIZTIT, TITLE, SIZLAB, XLBL, YLBL, IFONT, ISHAD
COMMON/AXES/IXS, XMN, XSTEP, XMX, XMULT, LOGX,
:                                IYS, YMN, YSTEP, YMX, YMULT, LOGY
COMMON/PLOTSIZE/XORIGIN, YORIGIN, XLENIN, YLENIN
COMMON/MESSAGE/NMESLNS, SIZMES, MESTXT, XMESF(MM), YMESF(MM),
:                                ANGMES(MM)
COMMON/INFO/MODIFIED
DIMENSION X(*), Y(*), TMPSCALE(2), NP(*), IPAKLEG(IPL), IPAKSTY(IPS)
EQUIVALENCE (STRTMP, LSTRTMP), (LLEGTIT, LEGTIT)
EQUIVALENCE (TITLE, LTITLE), (XLBL, LXLBL), (YLBL, LYLBL)
MODIFIED=.FALSE.
PAGEX=11.
PAGEY=8.5
IF(XTITF.LE.0.) XTITF=.5
IF(YTITF.LE.0.) YTITF=1.09
IF(XLEGF.LE.0.) XLEGF=.8
IF(YLEGF.LE.0.) YLEGF=.8
IF(XSTYF.LE.0.) XSTYF=.5
IF(YSTYF.LE.0.) YSTYF=1.08
IF(SIZSTY.LE.0.) SIZSTY=.15
IF(SIZLEG.LE.0.) SIZLEG=.15
IF(SIZMES.LE.0.) SIZMES=.15
IF(SIZTIT.LE.0.) SIZTIT=.3
IF(SIZLAB.LE.0.) SIZLAB=.3
IF(SIZSYM.LE.0.) SIZSYM=.5
IF(XMULT.LE.0.) XMULT=1.
IF(YMULT.LE.0.) YMULT=1.
IF(XORIGIN.LE.0.) XORIGIN=1.75
IF(YORIGIN.LE.0.) YORIGIN=1.25
IF(XLENIN.LE.0.) XLENIN=8.9
IF(YLENIN.LE.0.) YLENIN=6.0
NPT=0
DO 50 J=1, NCURV
```

```

50   NPT=NPT+NP (J)
      1 CONTINUE
      CALL GSSCGI
      CALL SPCMOD
      NPKWRD=LINEST (IPAKLEG, IPL, 60)
      NPKWRD=LINEST (IPAKSTY, IPS, 60)
      2 CONTINUE
      CALL NEWCLR (4HFORE)
      CALL SETDEV (0, 0)
      CALL SIMPLX
      CALL LINESP (2.5)
      IF (IFONT.EQ.1) CALL TRIPLX
      IF (IFONT.EQ.2) CALL SWISSL
      IF (IFONT.EQ.3) CALL SERIF
      IF (ISHAD.EQ.1) CALL SHDCHR (90., 1, .009, 1)
      CALL MX1ALF (8HSTANDARD, 1H{)
      CALL MX2ALF (6HL/CGRE, 1H{)
      CALL MX3ALF (5HGREEK, 1H~)
      CALL MX4ALF (5HMATHE, 1H\ )
      CALL MX5ALF (11HINSTRUCTION, 1H{)
      CALL MX6ALF (6HSPECIA, 1H{)
      CALL PHYSOR (XORIGIN, YORIGIN)
      CALL XREVTX
      CALL YREVTX
      CALL NOBRDR
      CALL YAXANG (0)
      CALL GRACE (0.)
      CALL NOCHEK
      CALL ALNMS (.5, .5)
      CALL THKRND (1.0)
      DO 51 J=1, NPT
      IF (XMULT.NE.0.) X(J)=X(J)*XMULT
51   IF (YMULT.NE.0.) Y(J)=Y(J)*YMULT
      CALL SCALE (X, NPT, AMN, ASTEP, AMX, IDECAB, DMNX, DMXX)
      IF (IXS.EQ.0) THEN
          XMN=AMN
          XSTEP=ASTEP
          XMK=AMX
      ENDIF
      IF (DMNX.LE.0.) LOGX=0
      CALL SCALE (Y, NPT, AMN, ASTEP, AMX, IDECOR, DMNY, DMXY)
      IF (IYS.EQ.0) THEN
          YMN=AMN
          YSTEP=ASTEP
          YMK=AMX
      ENDIF
      IF (DMNY.LE.0.) LOGY=0
      CALL PAGE (PAGEX, PAGEY)
      IF (NSTYLS.GT.0.AND.NSTYLS.LE.MS) THEN
          XSTYPOS=XLENIN*XSTYF
          YSTYPOS=YLENIN*YSTYF
          CALL ALNSTY (.5, .5)
          CALL HEIGHT (SIZSTY)
          CALL LINESP (2.0)
          DO 142 J=1, NSTYLS
          STRTMP=STYTXT (J)
          STRTMP (NUMCHR (STRTMP) +1: NUMCHR (STRTMP) +1) = ' $ '
          CALL LINES (LSTRTMP, IPAKSTY, J)

```

```

142  CONTINUE
      CALL LINE$P(2.5)
      ENDIF
      CALL AREA2D(XLENIN,YLENIN)
      LEG=0
      DO 144 J=1,NCURV
144  IF (NUMCHR(LEGTXT(J)).GT.0) LEG=1
      IF (LEG.EQ.1) THEN
          XLEGPOS=XLEGF*XLENIN
          YLEGPOS=YLEGF*YLENIN
          CALL ALNLEG(.5,.5)
          CALL HEIGHT(SIZLEG)
          NLEGTIT=NUMCHR(LEGTIT)
          IF (NLEGTIT.EQ.0) CALL MYLEGN(1H,1)
          IF (NLEGTIT.GT.0) CALL MYLEGN(LLEGTIT,NLEGTIT)
          DO 145 J=1,NCURV
          IF (NUMCHR(LEGTXT(J)).EQ.0) THEN
              CALL LINES(1H,IPAKLEG,J)
          ELSE
              STRTMP=LEGTXT(J)
              STRTMP(NUMCHR(STRTMP)+1:NUMCHR(STRTMP)+1)='$'
              CALL LINES(LSTRTMP,IPAKLEG,J)
          ENDIF
145  CONTINUE
      ENDIF
      NXCHR=NUMCHR(XLBL)
      IF (NXCHR.EQ.0) THEN
          XLBL=' '
          NXCHR=1
      ENDIF
      NYCHR=NUMCHR(YLBL)
      IF (NYCHR.EQ.0) THEN
          YLBL=' '
          NYCHR=1
      ENDIF
      IF (NUMCHR(TITLE).GT.0) THEN
          XMESPOS=XTITF*XLENIN
          YMESPOS=YTITF*YLENIN
          CALL HEIGHT(SIZTIT)
          CALL MESSAG(LTITLE,NUMCHR(TITLE),XMESPOS,YMESPOS)
      ENDIF
      CALL HEIGHT(SIZLAB)
      IF (LOGX.EQ.0.) CALL XNAME(LXLBL,NXCHR)
      IF (LOGY.EQ.0.) CALL YNAME(LYLBL,NYCHR)
      IF (IDECAB.LT.1) CALL XINTAX
      IF (IDECAB.GE.1) CALL RESET(6HXINTAX)
      IF (IDECOR.LT.1) CALL YINTAX
      IF (IDECOR.GE.1) CALL RESET(6HYINTAX)
      CALL XNMADJ(4HNONE)
      XTOIN=(XMX-XMN)/XLENIN
      YTOIN=(YMX-YMN)/YLENIN
      IF (LOGX.EQ.0 .AND. LOGY.EQ.0) THEN
          CALL XTICKS(2)
          CALL YTICKS(2)
          CALL GRAF(XMN,XSTEP,XMX,YMN,YSTEP,YMX)
          CALL RESET(6HXREVTK)
          CALL RESET(6HYREVTK)
          CALL XNONUM

```

```

CALL YNONUM
CALL XGRAXS (XMN,XSTEP,XMX,XLENIN,1H ,1,0.,YLENIN)
CALL YGRAXS (YMN,YSTEP,YMX,YLENIN,1H ,1,XLENIN,0.)
ELSEIF (LOGX.EQ.0 .AND. LOGY.EQ.1) THEN
CALL XTICKS(2)
CALL YTICKS(1)
CALL GRAF (XMN,XSTEP,XMX,YMN,YSTEP,YMX)
* calculate cycles even if not used to provide timing delay for axis labels
CYCLES=ALOG10 (DMKY/DMNY)
FAC=CYCLES*.5+1.
IF (IYS.EQ.0.OR.XMN.LE.0.OR.YMX.LE.0) THEN
    YMN = DMNY/FAC
    YMX = DMKY*FAC
ENDIF
YCYCLE = YLENIN / ALOG10 (YMX/YMN)
A=ALOG10 (YMN)
B=IFIX(A)
POWER10 = (ABS(A-B) .LT.1.E-6)
IF (.NOT.POWER10) CALL YAXEND (7HNOFIRST)
CALL YLGAXS (YMN,YCYCLE,YLENIN,LXLBL,NYCHR,0.,0.)
CALL RESET (6HYAXEND)
CALL RESET (6HXREVTK)
CALL RESET (6HYREVTK)
CALL XNONUM
CALL YNONUM
CALL XGRAXS (XMN,XSTEP,XMX,XLENIN,1H ,1,0.,YLENIN)
CALL YLGAXS (YMN,YCYCLE,YLENIN,1H ,1,XLENIN,0.)
ELSEIF (LOGX.EQ.1 .AND. LOGY.EQ.0) THEN
CALL XTICKS(1)
CALL YTICKS(2)
CALL GRAF (XMN,XSTEP,XMX,YMN,YSTEP,YMX)
* calculate cycles even if not used to provide timing delay for axis labels
CYCLES=ALOG10 (DMCX/DMNX)
FAC=CYCLES*.5+1.
IF (IXS.EQ.0.OR.XMN.LE.0.OR.XMX.LE.0) THEN
    XMN = DMNX/FAC
    XMX = DMCX*FAC
ENDIF
XCYLE = XLENIN / ALOG10 (XMX/XMN)
A=ALOG10 (XMN)
B=IFIX(A)
POWER10 = (ABS(A-B) .LT.1.E-6)
IF (.NOT.POWER10) CALL XAXEND (7HNOFIRST)
CALL XLGAXS (XMN,XCYCLE,XLENIN,LXLBL,NXCHR,0.,0.)
CALL RESET (6HXAXEND)
CALL RESET (6HXREVTK)
CALL RESET (6HYREVTK)
CALL XNONUM
CALL YNONUM
CALL XLGAXS (XMN,XCYCLE,XLENIN,1H ,1,0.,YLENIN)
CALL YGRAXS (YMN,YSTEP,YMX,YLENIN,1H ,1,XLENIN,0.)
ELSEIF (LOGX.EQ.1 .AND. LOGY.EQ.1) THEN
CALL XTICKS(1)
CALL YTICKS(1)
CALL GRAF (XMN,XSTEP,XMX,YMN,YSTEP,YMX)
* calculate cycles even if not used to provide timing delay for axis labels
CYCLES=ALOG10 (DMCX/DMNX)
FAC=CYCLES*.5+1.

```

```

IF (IXS.EQ.0.OR.XMN.LE.0.OR.XMX.LE.0) THEN
  XMN = DMNX/FAC
  XMX = DMXX*FAC
ENDIF
XCYLE = XLENIN / ALOG10(XMX/XMN)
A=ALOG10(XMN)
B=IFIX(A)
POWER10 = (ABS(A-B).LT.1.E-6)
IF(.NOT.POWER10) CALL XAXEND(7HNOFIRST)
CALL XREVTX
CALL RESET(6HXNONUM)
CALL XLGAXS(XMN,XCYLE,XLENIN,LXLBL,NXCHR,0.,0.)
CALL RESET(6HXAXEND)
CALL RESET(6HXREVTX)
CALL XNONUM
CALL XLGAXS(XMN,XCYLE,XLENIN,1H,1,0.,YLENIN)
* calculate cycles even if not used to provide timing delay for axis labels
CYCLES=ALOG10(DMKY/DMNY)
FAC=CYCLES*.5+1.
IF (IYS.EQ.0.OR.YMN.LE.0.OR.YMX.LE.0) THEN
  YMN = DMNY/FAC
  YMX = DMKY*FAC
ENDIF
YCYLE = YLENIN / ALOG10(YMX/YMN)
A=ALOG10(YMN)
B=IFIX(A)
POWER10 = (ABS(A-B).LT.1.E-6)
IF(.NOT.POWER10) CALL YAXEND(7HNOFIRST)
CALL YREVTX
CALL RESET(6HYNONUM)
CALL YLGAXS(YMN,YCYLE,YLENIN,LYLBL,NYCHR,0.,0.)
CALL RESET(6HYAXEND)
CALL RESET(6HYREVTX)
CALL YNONUM
CALL YLGAXS(YMN,YCYLE,YLENIN,1H,1,XLENIN,0.)
ENDIF
CALL XREVTX
CALL YREVTX
CALL RESET(6HXNONUM)
CALL RESET(6HYNONUM)
CALL THKFRM(.03)
CALL FRAME
CALL SCLPIC(SIZSYM/SIZLAB)
XMININ=-XPOSN(4HLEFT,5HLOWER)
YMININ=-YPOSN(4HLEFT,5HLOWER)
NPKNT=0
CALL SETSAT(4HVIVI)
CALL SETINT(4HBRIL)
DO 82 M=1,NCURV
  IBEG=NPKNT+1
  NPKNT=NPKNT+NP(M)
  IF(LSKIP(M).EQ.0) LSKIP(M)=MAX(1,NP(M)/25)
  CALL MARKER(LSYMBOL(M))
  IF(LBOLD(M).EQ.1) CALL THKCRV(.03)
  CALL NEWCLR(4HFORE)
  IF(LCOLOR(M).EQ.1) CALL NEWCLR(3HRED)
  IF(LCOLOR(M).EQ.2) CALL NEWCLR(5HGREEN)
  IF(LCOLOR(M).EQ.3) CALL NEWCLR(4HBLUE)

```

```

IF (LCOLOR(M).EQ.4) CALL NEWCLR(7HMAGENTA)
IF (LCOLOR(M).EQ.5) CALL NEWCLR(6HYELLOW)
IF (LCOLOR(M).EQ.6) CALL NEWCLR(4HCYAN)
IF (LCOLOR(M).EQ.7) CALL NEWCLR(5HWHITE)
IF (LTYPE(M).EQ.-2) CALL CURVE(X(IBEG),Y(IBEG),NP(M),-LSKIP(M))
IF (LTYPE(M).EQ.-1) CALL CURVE(X(IBEG),Y(IBEG),NP(M),LSKIP(M))
IF (LTYPE(M).EQ.1) CALL DASH
IF (LTYPE(M).EQ.2) CALL DOT
IF (LTYPE(M).EQ.3) CALL CHNDSH
IF (LTYPE(M).EQ.4) CALL CHNDOT
IF (LTYPE(M).GT.-1.AND.LTYPE(M).LT.5) THEN
  CALL LEGLIN
  CALL CURVE(X(IBEG),Y(IBEG),NP(M),0)
ENDIF
IF (NUMCHR(LEGTXT(M)).EQ.0) CALL DELLEG(M)
CALL RESET(4HDASH)
CALL RESET(3HDOT)
CALL RESET(6HCHNDSH)
CALL RESET(6HCHNDOT)
CALL RESET(6HCHKCRV)
CALL RESET(6HLEGLIN)
82 CONTINUE
CALL NEWCLR(4HFORE)
IF (NSTYLNS.GT.0) THEN
  CALL LSTORY(IPAKSTY,NSTYLNS,XSTYPOS,YSTYPOS)
ENDIF
IF (NMESLNS.GT.0) THEN
  DO 83 J=1,NMESLNS
    XMESPOS=XMESF(J)*XLENIN
    YMESPOS=YMESF(J)*YLENIN
    CALL ALNMES(.5,.5)
    CALL HEIGHT(SIZMES)
    CALL ANGLE(ANGMES(J))
    STRTMP=MESTXT(J)
    NMES=NUMCHR(STRTMP)
    CALL MESSAG(LSTRTMP,NMES,XMESPOS,YMESPOS)
83 CONTINUE
  ENDF
  IF (LEG.EQ.1) THEN
    CALL HEIGHT(SIZLEG)
    CALL LEGEND(IPAKLEG,NCURV,XLEGPOS,YLEGPOS)
  ENDF
  CALL ENDPL(0)
  CALL DONEPL
  DO 52 J=1,NPT
    IF (XMULT.NE.0.) X(J)=X(J)/XMULT
52 IF (YMULT.NE.0.) Y(J)=Y(J)/YMULT
61 CONTINUE
  WRITE(*,(' ' Plot Code [Redraw]: ' '))
  IF (NCHRR(ALINE).EQ.0) GO TO 1
  LET=ALINE(1:NUMCHR(ALINE)-1)
  DO 53 I=1,NUMCHR(LET)
    ICHR = ICHAR(LET(I:I))
53 IF (ICHR.GT.96.AND.ICHR.LT.123) LET(I:I)=CHAR(ICHR-32)
    IF (LET.EQ.'E'.OR.LET.EQ.'Q') GO TO 63
    IF (LET.EQ.'H') THEN
      PRINT *, 'Enter one of the following codes:'
      PRINT *, 'AM create/modify annotation message lines'

```

```

PRINT *, 'AS create/modify annotation story'
PRINT *, 'E (or Q) exit'
PRINT *, 'F cycle through fonts SIMPLX TRIPLX SWISSL and SERIF'
PRINT *, 'H help'
PRINT *, 'LB modify line LBOLD'
PRINT *, 'LC modify line LCOLOR'
PRINT *, 'LE create/modify legend'
PRINT *, 'LS modify line symbols'
PRINT *, 'LT modify line LTYPE'
PRINT *, 'LX toggle linear/log x-axis'
PRINT *, 'LY toggle linear/log y-axis'
PRINT *, 'PS modify plot size or position'
PRINT *, 'SH toggle shading of SWISSL and SERIF fonts'
PRINT *, 'T modify plot title'
PRINT *, 'XL modify x label'
PRINT *,
:      'XS modify x min, max(autoscale if = min), step, multiplier'
PRINT *, 'YL modify y label'
PRINT *,
:      'YS modify y min, max(autoscale if = min), step, multiplier'
PRINT *
GO TO 61
ENDIF
MODIFIED=.TRUE.
IF (LET.EQ.'F') IFONT=MOD(IFONT+1,4)
IF (LET.EQ.'SH') ISHAD=IABS(ISHAD-1)
IF (LET.EQ.'LB') THEN
  STRTMP='('' Line Bold ['',10I3,'']: '')'
  WRITE (STRTMP(17:18),'(I2)') NCURV
  WRITE (*,STRTMP) (LBOLD(J),J=1,NCURV)
  IF (NCHRR(ALINE).GT.0) READ(99,*) (LBOLD(J),J=1,NCURV)
ENDIF
IF (LET.EQ.'LC') THEN
  STRTMP='('' Line Color ['',10I2,'']: '')'
  WRITE (STRTMP(18:19),'(I2)') NCURV
  WRITE (*,STRTMP) (LCOLOR(J),J=1,NCURV)
  IF (NCHRR(ALINE).GT.0) READ(99,*) (LCOLOR(J),J=1,NCURV)
ENDIF
IF (LET.EQ.'LS') THEN
  STRTMP='('' Symbol # ['',10I3,'']: '')'
  WRITE (STRTMP(16:17),'(I2)') NCURV
  WRITE (*,STRTMP) (LSYMBOL(J),J=1,NCURV)
  IF (NCHRR(ALINE).GT.0) READ(99,*) (LSYMBOL(J),J=1,NCURV)
  STRTMP='('' Symbol Skip ['',10I4,'']: '')'
  WRITE (STRTMP(19:20),'(I2)') NCURV
  WRITE (*,STRTMP) (LSKIP(J),J=1,NCURV)
  IF (NCHRR(ALINE).GT.0) READ(99,*) (LSKIP(J),J=1,NCURV)
  WRITE (*,('' Symbol Size ['',F4.2,'']: '')) SIZSYM
  IF (NCHRR(ALINE).GT.0) READ(99,*) SIZSYM
ENDIF
IF (LET.EQ.'LT') THEN
  STRTMP='('' Line Type ['',10I3,'']: '')'
  WRITE (STRTMP(17:18),'(I2)') NCURV
  WRITE (*,STRTMP) (LTYPE(J),J=1,NCURV)
  IF (NCHRR(ALINE).GT.0) READ(99,*) (LTYPE(J),J=1,NCURV)
ENDIF
IF (LET.EQ.'XL') THEN
  WRITE (*,('' X Label ['',A,'']/10('' ''))) XLBL

```

```

      IF (NCHRR (ALINE) .GT. 0) XLBL=ALINE (1:NUMCHR (ALINE) -1)
      WRITE (*, ' ('' Chrsiz ['', F4.2, '']: ')) SIZLAB
      IF (NCHRR (ALINE) .GT. 0) READ (99, *) SIZLAB
ENDIF
IF (LET.EQ. 'XS') THEN
  XMULTOLD=XMULT
  WRITE (*, ' ('' Xmin, Xmax, Xstep, Xmultiplier ['', 1P4E10.2,
: '']: ')/') XMN, XMX, XSTEP, XMULT
  IF (NCHRR (ALINE) .GT. 0) READ (99, *) XMN, XMX, XSTEP, XMULT
  IF (XMULT.NE.XMULTOLD.AND.NXCHR.GT.1)
: XLBL=XLBL (1:NUMCHR (XLBL))// ' ?'
  XMULTOLD=XMULT
  IF (XMN.EQ.XMX) THEN
    IXS=0
  ELSE
    TMPSCALE (1)=XMN
    TMPSCALE (2)=XMX
    CALL SCALE (TMPSCALE, 2, DUM1, DUM2, DUM3, IDECAB, DUM4, DUM5)
    IXS=1
  ENDIF
ENDIF
ENDIF
IF (LET.EQ. 'LX') THEN
  CALL SCALE (X, NPT, XMN, XSTEP, XMX, IDUM2, DMNX, DMXX)
  IF (DMNX.LE.0.0) THEN
    PRINT *, 'Xmin must be 0.0 ... LX cancelled ... '
  ELSE
    LOGX=IABS (LOGX-1)
  ENDIF
ENDIF
ENDIF
IF (LET.EQ. 'YL') THEN
  WRITE (*, ' ('' Y Label ['', A, '']/10 ('' '))') YLBL
  IF (NCHRR (ALINE) .GT. 0) YLBL=ALINE (1:NUMCHR (ALINE) -1)
  WRITE (*, ' ('' Chrsiz ['', F4.2, '']: ')) SIZLAB
  IF (NCHRR (ALINE) .GT. 0) READ (99, *) SIZLAB
ENDIF
IF (LET.EQ. 'YS') THEN
  YMULTOLD=YMULT
  WRITE (*, ' ('' Ymin, Ymax, Ystep, Ymultiplier ['', 1P4E10.2,
: '']: ')/') YMN, YMX, YSTEP, YMULT
  IF (NCHRR (ALINE) .GT. 0) READ (99, *) YMN, YMX, YSTEP, YMULT
  IF (YMULT.NE.YMULTOLD.AND.NYCHR.GT.1)
: YLBL=YLBL (1:NUMCHR (YLBL))// ' ?'
  YMULTOLD=YMULT
  IF (YMN.EQ.YMX) THEN
    IYS=0
  ELSE
    TMPSCALE (1)=YMN
    TMPSCALE (2)=YMX
    CALL SCALE (TMPSCALE, 2, DUM1, DUM2, DUM3, IDECOR, DUM4, DUM5)
    IYS=1
  ENDIF
ENDIF
ENDIF
IF (LET.EQ. 'LY') THEN
  CALL SCALE (Y, NPT, YMN, YSTEP, YMX, IDUM2, DMNY, DMXY)
  IF (DMNY.LE.0.0) THEN
    PRINT *, 'Ymin must be 0.0 ... LY cancelled ... '
  ELSE
    LOGY=IABS (LOGY-1)

```

```

        ENDIF
ENDIF
IF (LET.EQ.'T') THEN
    WRITE(*,'(' Title [' ',A,'']'/8(' ' '))') TITLE
    IF (NCHRR(ALINE).GT.0) TITLE=ALINE(1:NUMCHR(ALINE)-1)
    WRITE(*,'(' Chrsiz [' ',F4.2,'']: ' ')) SIZTIT
    IF (NCHRR(ALINE).GT.0) READ(99,*) SIZTIT
    WRITE(*,'(' Title Fractional XPOS,YPOS [' ',
: F4.2,F6.2,'']: ' ')) XTITF,YTITF
    IF (NCHRR(ALINE).GT.0) READ(99,*) XTITF,YTITF
ENDIF
IF (LET.EQ.'LE') THEN
    WRITE(*,'(' Legend Title [' ',A,'']'/15(' ' '))') LEGTIT
    IF (NCHRR(ALINE).GT.0) LEGTIT=ALINE(1:NUMCHR(ALINE)-1)
    DO 71 J=1,NCURV
    WRITE(*,'(' Line ' ',I2,' [' ',A,'']'/10(' ' '))')
: J,LEGTXT(J)
    IF (NCHRR(ALINE).GT.0) LEGTXT(J)=ALINE(1:NUMCHR(ALINE)-1)
71 CONTINUE
    WRITE(*,'(' Chrsiz [' ',F4.2,'']: ' ')) SIZLEG
    IF (NCHRR(ALINE).GT.0) READ(99,*) SIZLEG
    WRITE(*,'(' Legend Fractional XPOS,YPOS [' ',
: F4.2,F6.2,'']: ' ')) XLEGF,YLEGF
    IF (NCHRR(ALINE).GT.0) READ(99,*) XLEGF,YLEGF
ENDIF
IF (LET.EQ.'AM') THEN
    WRITE(*,'(' # Lines [' ',I2,'']: ' ')) NMESLNS
    IF (NCHRR(ALINE).GT.0) READ(99,*) NMESLNS
    IF (NMESLNS.GT.NM) NMESLNS=NM
    IF (NMESLNS.GT.0) THEN
        DO 12 J=1,NMESLNS
        WRITE(*,'(' Line ' ',I2,' [' ',A,'']'/10(' ' '))') J,MESTXT(J)
        IF (NCHRR(ALINE).GT.0) MESTXT(J)=ALINE(1:NUMCHR(ALINE)-1)
        WRITE(*,'(' Fractional XPOS,YPOS,ANG [' ',2F6.2,
: F6.1,'']: ' ')) XMESF(J),YMESF(J),ANGMES(J)
        IF (NCHRR(ALINE).GT.0) READ(99,*) XMESF(J),YMESF(J),ANGMES(J)
12 CONTINUE
        WRITE(*,'(' Chrsiz [' ',F4.2,'']: ' ')) SIZMES
        IF (NCHRR(ALINE).GT.0) READ(99,*) SIZMES
    ENDIF
ENDIF
IF (LET.EQ.'AS') THEN
    WRITE(*,'(' # Lines [' ',I2,'']: ' ')) NSTYLNS
    IF (NCHRR(ALINE).GT.0) READ(99,*) NSTYLNS
    IF (NSTYLNS.GT.0) THEN
        DO 14 J=1,NSTYLNS
        WRITE(*,'(' Line ' ',I2,' [' ',A,'']'/10(' ' '))') J,STYTXT(J)
        IF (NCHRR(ALINE).GT.0) STYTXT(J)=ALINE(1:NUMCHR(ALINE)-1)
14 CONTINUE
        WRITE(*,'(' Chrsiz [' ',F4.2,'']: ' ')) SIZSTY
        IF (NCHRR(ALINE).GT.0) READ(99,*) SIZSTY
        WRITE(*,'(' Annotation Fractional XPOS,YPOS [' ',
: 2F6.3,'']: ' ')) XSTYF,YSTYF
        IF (NCHRR(ALINE).GT.0) READ(99,*) XSTYF,YSTYF
    ENDIF
ENDIF
IF (LET.EQ.'PS') THEN
    WRITE(*,'(' Xorigin, Yorigin, Xlength, Ylength [' ',4F6.2,

```

```

:      '']: '/'') XORIGIN,YORIGIN,XLENIN,YLENIN
      IF (NCHRR (ALINE) .GT.0) READ (99,*) XORIGIN,YORIGIN,XLENIN,YLENIN
ENDIF
GO TO 61
63 CONTINUE
RETURN
END

```

```

SUBROUTINE SCALE (DATA,NP,SMIN,SSTP,SMAX,NDEC,DMIN,DMAX)
DIMENSION DATA (NP)
DMIN=DATA (1)
DMAX=DATA (1)
DO 10 J=2,NP
  DMIN=MIN (DMIN,DATA (J))
10 DMAX=MAX (DMAX,DATA (J))
  DMN=DMIN
  DMX=DMAX
  IF (DMIN.EQ.DMAX) THEN
    DMN=DMN-1.
    DMX=DMX+1.
  ENDIF
  DRNG=ABS (DMX-DMN)
  DO 11 J=1,60
    FAC=10.** (J-31)
    IF ((DRNG*FAC).GE.3.9) GO TO 12
11 CONTINUE
12 CONTINUE
  SSTP=1./FAC
  CALL LINSSCALE (SMIN,SMAX,DMN,DMX,SSTP,NLINES)
  IF (NLINES.LE.8) GO TO 40
  SSTP=2./FAC
  CALL LINSSCALE (SMIN,SMAX,DMN,DMX,SSTP,NLINES)
  IF (NLINES.LE.8) GO TO 40
  SSTP=5./FAC
  CALL LINSSCALE (SMIN,SMAX,DMN,DMX,SSTP,NLINES)
  IF (NLINES.LE.8) GO TO 40
  FAC=FAC*.1
  GO TO 12
40 CONTINUE
  IF (FAC.LT.5.) NDEC=0
  IF (FAC.GT.5.) NDEC=ALOG10 (FAC)+.5
  RETURN
END

```

```

SUBROUTINE LINSSCALE (SMIN,SMAX,DMN,DMX,SSTP,NLINES)
SMIN=IFIX (DMN/SSTP+.5)*SSTP
SMAX=IFIX (DMX/SSTP+.5)*SSTP
C   IF (SMIN.GT.DMN- (SMAX-SMIN)*.04) SMIN=SMIN-SSTP
C   IF (SMIN.GT.DMN- (SMAX-SMIN)*.04) SMIN=SMIN-SSTP
C   IF (SMAX.LT.DMX+ (SMAX-SMIN)*.04) SMAX=SMAX+SSTP
C   IF (SMAX.LT.DMX+ (SMAX-SMIN)*.04) SMAX=SMAX+SSTP
IF (SMIN.GT.DMN) SMIN=SMIN-SSTP
IF (SMIN.GT.DMN) SMIN=SMIN-SSTP
IF (SMAX.LT.DMX) SMAX=SMAX+SSTP
IF (SMAX.LT.DMX) SMAX=SMAX+SSTP
NLINES= (SMAX-SMIN)/SSTP+.5
RETURN
END

```

```

FUNCTION NCHRR (ALINE)
CHARACTER ALINE* (*)
READ ' (A)', ALINE
NCHRR=NUMCHR (ALINE)
IF (NCHRR.GT.0) THEN
  ALINE (NCHRR+1:NCHRR+1)='/'
  IF (ALINE.EQ. '/' '/') ALINE='  '
  REWIND 99
  WRITE (99,' (A)') ALINE (1:NCHRR+1)
  REWIND 99
ENDIF
RETURN
END

```

```

FUNCTION NUMCHR (STRING)
CHARACTER STRING* (*), BLNK, NUL
BLNK=' '
NUL=CHAR (0)
MAXCHR=LEN (STRING)
11 DO 20 NUMCHR=MAXCHR,1,-1
  IF (STRING (NUMCHR:NUMCHR) .NE. BLNK.AND. STRING (NUMCHR:NUMCHR) .NE. NUL)
    :RETURN
20 CONTINUE
RETURN
END

```

```

SUBROUTINE MYSPEC (IENTRY)
PARAMETER (MC=50)
COMMON/LINES/LBOLD (MC), LCOLOR (MC), LTYPE (MC),
: LSYMBOL (MC), LSKIP (MC), SIZSYM
IF (LCOLOR (IENTRY) .EQ. 0) CALL NEWCLR (4HFORE)
IF (LCOLOR (IENTRY) .EQ. 1) CALL NEWCLR (3HRED)
IF (LCOLOR (IENTRY) .EQ. 2) CALL NEWCLR (5HGREEN)
IF (LCOLOR (IENTRY) .EQ. 3) CALL NEWCLR (4HBLUE)
IF (LCOLOR (IENTRY) .EQ. 4) CALL NEWCLR (7HMAGENTA)
IF (LCOLOR (IENTRY) .EQ. 5) CALL NEWCLR (6HYELLOW)
IF (LCOLOR (IENTRY) .EQ. 6) CALL NEWCLR (4HCYAN)
IF (LCOLOR (IENTRY) .EQ. 7) CALL NEWCLR (5HWHITE)
RETURN
END

```

Appendix B

VAX Version Of TPLOT

```
*****
*          VAX FORTRAN Version of TPLOT Source Code as of January 1991          *
*                                                                                   *
*      MC = max curves    MS = max story lines    MM = max messages              *
*      IPL = legend packing array size    IPS = story packing array size        *
*****

SUBROUTINE TPLOT (NCURV,X,Y,NP,TT)
  CHARACTER LET*2, TERMTYPE*3, TT*3, VMSTERM*3, ALINE*132, FILE*40
  CHARACTER OUTFILE*40
  LOGICAL POWER10, MODIFIED, DISPLAY
  CHARACTER*60 TITLE,XLBL,YLBL,LEGTIT*20,LEGTXT(50),STYTXT(50),MESTXT(50)
  COMMON/PLOTSTORY/NSTYLSN,SIZSTY,STYTXT,XSTYF,YSTYF
  COMMON/PLOTLEGEND/XLEGF,YLEGF,SIZLEG,LEGTIT,LEGTXT
  COMMON/PLOTLINES/LBOLD(50),LCOLOR(50),LTYPE(50),
:  LSYMBOL(50),LSKIP(50),SIZSYM
  COMMON/PLOTLABEL/XTITF,YTITF,SIZTIT,TITLE,SIZLAB,XLBL,YLBL,IFONT,ISHAD
  COMMON/PLOTAXES/IXS,XMN,XSTEP,XMX,XMULT,LOGX,
:  IYS,YMN,YSTEP,YMX,YMULT,LOGY
  COMMON/PLOTSIZE/XORIGIN,YORIGIN,XLENIN,YLENIN
  COMMON/MESSAGE/NMESLNS,SIZMES,MESTXT,XMESF(50),YMESF(50),ANGMES(50)
  COMMON/INFO/MODIFIED,FILE
  DIMENSION X(*),Y(*),NP(*)
  DATA LSYMBOL/0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,31*0/
  DATA XLEGF/.8/,YLEGF/.8/,XSTYF/.5/,YSTYF/1.08/,XTITF/.5/,YTITF/1.09/
  DATA SIZMES/.15/,MESTXT/50*' '/
  DATA XORIGIN/1.75/,YORIGIN/1.25/,XLENIN/8.9/,YLENIN/6./
  DATA SIZSYM/0.5/, SIZLAB/0.25/, SIZSTY/0.15/
  DATA SIZLEG/0.15/, LEGTIT/' '/, LEGTXT/50*' '/, STYTXT/50*' '/
  DATA SIZTIT/0.3/, TITLE/' '/, XLBL/' '/, YLBL/' '/, XMULT/1/, YMULT/1/
  MODIFIED=.FALSE.
  IF (SIZMES.LE.0.) SIZMES=.15
  IF (SIZSYM.LE.0.) SIZSYM=.6
  IF (SIZLAB.LE.0.) SIZLAB=.25
  IF (SIZSTY.LE.0.) SIZSTY=.15
  IF (SIZLEG.LE.0.) SIZLEG=.15
  IF (SIZTIT.LE.0.) SIZTIT=.3
  IF (TT.NE.'TEK'.AND.TT.NE.'REG'.AND.TT.NE.'340'.AND.TT.NE.'240') THEN
    TERMTYPE=VMSTERM()
  ELSE
    TERMTYPE=TT
  ENDIF
  PAGEX=11.
  PAGEY=8.5
  NPT=0
  DO 50 J=1,NCURV
50  NPT=NPT+NP(J)
1  CONTINUE
  IF (TERMTYPE.EQ.'REG') THEN
    CALL REGIS(3,1)
  ELSEIF (TERMTYPE.EQ.'340') THEN
    CALL VT3XXO(1,340)
```

```

ELSEIF (TERMTYPE.EQ.'240') THEN
  CALL VT240
ELSE
  CALL TEKALL(4010,960,0,0,0)
ENDIF
DISPLAY=.TRUE.
2 CONTINUE
CALL HWSCL('SCREEN')
CALL NEWCLR('FORE')
CALL SETDEV(0,0) ! no summary messages
CALL SIMPLX
IF (IFONT.EQ.1) CALL TRIPLX
IF (IFONT.EQ.2) CALL SWISSL
IF (IFONT.EQ.3) CALL SERIF
IF (ISHAD.EQ.1) CALL SHDCHR(90.,1,.009,1)
CALL MX1ALF('STANDARD',' ']')
CALL MX2ALF('L/CGRE','['')
CALL MX3ALF('GREEK','~')
CALL MX4ALF('MATHE','\')
CALL MX5ALF('INSTRUCTION','(')
CALL MX6ALF('SPECIA','|')
CALL PHYSOR(XORIGIN,YORIGIN)
CALL LINESP(2.5)
CALL HEIGHT(SIZLAB)
CALL XREVTX
CALL YREVTX
CALL NOBRDR
CALL YAXANG(0)
CALL GRACE(0.)
CALL NOCHK
CALL ALNME(.5,.5)
CALL THKRD(1.0) ! TO PREVENT SPIKING ON BOLDED CURVES
DO 51 J=1,NPT
IF (XMULT.NE.0.) X(J)=X(J)*XMULT
51 IF (YMULT.NE.0.) Y(J)=Y(J)*YMULT
CALL SCALE(X,NPT,AMN,ASTEP,AMX,DMNX,DMXX)
IF (IXS.EQ.0) THEN
  XMN=AMN
  XSTEP=ASTEP
  XMX=AMX
ENDIF
IF (DMNX.LE.0.) LOGX=0
CALL SCALE(Y,NPT,AMN,ASTEP,AMX,DMNY,DMY)
IF (IYS.EQ.0) THEN
  YMN=AMN
  YSTEP=ASTEP
  YMX=AMX
ENDIF
IF (DMNY.LE.0.) LOGY=0
IF (TERMTYPE.NE.'REG'.AND.TERMTYPE.NE.'340'.AND.TERMTYPE.NE.'240'
: .AND.DISPLAY) THEN
C --- (ESC)^L --- switch kermit to 4010 mode and clear screen
  WRITE(*,'('$',80A1)') 27,12 !
C --- (ESC)[0;1;33;44m --- set kermit 4010 mode colors to yellow on blue
  WRITE(*,'('$',80A1)') 27,91,48,59,49,59,51,51,59,52,52,109
ENDIF
CALL PAGE(PAGEX,PAGEY)
* call lines once regardless of NSTYLNS to assign a unique number to IPAKSTY

```

```

* and avoid mixing story and legend information.
CALL LINES(' ',IPAKSTY,1)
IF (NSTYLS.GT.0) THEN                                ! -- if story exists... --
    XSTYPOS=XSTYF*XLENIN
    YSTYPOS=YSTYF*YLENIN
    CALL ALNSTY(.5,.5)
    CALL HEIGHT(SIZSTY)
    CALL LINESP(2.0)
    DO 142 J=1,NSTYLS
    IF (NUMCHR(STYTXT(J)).EQ.0) CALL LINES(' ',IPAKSTY,J)
    IF (NUMCHR(STYTXT(J)).GT.0) CALL LINES(STYTXT(J),IPAKSTY,J)
142  CONTINUE
    CALL LINESP(2.5)
ENDIF
CALL AREA2D(XLENIN,YLENIN)
LEG=0
* call lines once regardless of legend status to assign a unique number to
* IPAKLEG and avoid mixing story and legend information.
CALL LINES(' ',IPAKLEG,1)
DO 144 J=1,NCURV
144  IF (NUMCHR(LEGTXT(J)).GT.0) LEG=1
    IF (LEG.EQ.1) THEN                                ! -- if legend exists... --
        XLEGPOS=XLEGF*XLENIN
        YLEGPOS=YLEGF*YLENIN
        CALL ALNLEG(0.5,0.5)
        CALL HEIGHT(SIZLEG)
        NLEGTIT=NUMCHR(LEGTIT)
        IF (NLEGTIT.EQ.0) CALL LEGNAM(' ',1)
        IF (NLEGTIT.GT.0) CALL LEGNAM(LEGTIT,NLEGTIT)
        DO 145 J=1,NCURV
        IF (NUMCHR(LEGTXT(J)).EQ.0) CALL LINES(' ',IPAKLEG,J)
145  IF (NUMCHR(LEGTXT(J)).GT.0) CALL LINES(LEGTXT(J),IPAKLEG,J)
    ENDIF
    NXCHR=NUMCHR(XLBL)
    IF (NXCHR.EQ.0) THEN
        XLBL=' '
        NXCHR=1
    ENDIF
    NYCHR=NUMCHR(YLBL)
    IF (NYCHR.EQ.0) THEN
        YLBL=' '
        NYCHR=1
    ENDIF
    NTCHR=NUMCHR(TITLE)
    IF (NTCHR.GT.0) THEN
        XMESPOS=XTITF*XLENIN
        YMESPOS=YTITF*YLENIN
        CALL HEIGHT(SIZTIT)
        CALL MESSAG(TITLE,NTCHR,XMESPOS,YMESPOS)
    ENDIF
    CALL HEIGHT(SIZLAB)
    IF (LOGX.EQ.0.) CALL XNAME(XLBL,NXCHR)
    IF (LOGY.EQ.0.) CALL YNAME(YLBL,NYCHR)
    IF (XMN.EQ.AINT(XMN).AND.XSTEP.EQ.AINT(XSTEP)) THEN
        CALL XINTAX
    ELSE
        CALL RESET('XINTAX')
    ENDIF

```

```

IF (YMN.EQ.AINT(YMN).AND.YSTEP.EQ.AINT(ystep)) THEN
  CALL YINTAX
ELSE
  CALL RESET('YINTAX')
ENDIF
CALL XNMADJ('NONE')
XTOIN=(XMX-XMN)/XLENIN
YTOIN=(YMX-YMN)/YLENIN
IF (LOGX.EQ.0 .AND. LOGY.EQ.0) THEN      ! --linlin--
  CALL XTICKS(2)
  CALL YTICKS(2)
  CALL GRAF(XMN,XSTEP,XMX,YMN,ystep,YMX)
  CALL RESET('XREVTX')
  CALL RESET('YREVTX')
  CALL XNONUM
  CALL YNONUM
  CALL XGRAXS(XMN,XSTEP,XMX,XLENIN,' ',1,0.,YLENIN)
  CALL YGRAXS(YMN,ystep,YMX,YLENIN,' ',1,XLENIN,0.)
ELSEIF (LOGX.EQ.0 .AND. LOGY.EQ.1) THEN  ! --linlog--
  CALL XTICKS(2)
  CALL YTICKS(1)
  CALL GRAF(XMN,XSTEP,XMX,YMN,ystep,YMX)
  IF (IYS.EQ.0.OR.YMN.LE.0.OR.YMX.LE.0) CALL LOGSCALE(YMN,YMX,DMNY,DMXY)
  YCYCLE = YLENIN / ALOG10(XMX/YMN)
  IF (.NOT.POWER10(YMN).AND.POWER10(YMX)) CALL YAXEND('NOFIRST')
  IF (POWER10(YMN).AND..NOT.POWER10(YMX)) CALL YAXEND('NOLAST')
  IF (.NOT.POWER10(YMN).AND..NOT.POWER10(YMX)) CALL YAXEND('NOENDS')
  CALL YLGAXS(YMN,ycycle,YLENIN,YLBL,NYCHR,0.,0.)
  CALL RESET('YAXEND')
  CALL RESET('XREVTX')
  CALL RESET('YREVTX')
  CALL XNONUM
  CALL YNONUM
  CALL XGRAXS(XMN,XSTEP,XMX,XLENIN,' ',1,0.,YLENIN)
  CALL YLGAXS(YMN,ycycle,YLENIN,' ',1,XLENIN,0.)
ELSEIF (LOGX.EQ.1 .AND. LOGY.EQ.0) THEN  ! --loglin--
  CALL XTICKS(1)
  CALL YTICKS(2)
  CALL GRAF(XMN,XSTEP,XMX,YMN,ystep,YMX)
  IF (IXS.EQ.0.OR.XMN.LE.0.OR.XMX.LE.0) CALL LOGSCALE(XMN,XMX,DMNX,DMXX)
  XCYCLE = XLENIN / ALOG10(XMX/XMN)
  IF (.NOT.POWER10(XMN).AND.POWER10(XMX)) CALL XAXEND('NOFIRST')
  IF (POWER10(XMN).AND..NOT.POWER10(XMX)) CALL XAXEND('NOLAST')
  IF (.NOT.POWER10(XMN).AND..NOT.POWER10(XMX)) CALL XAXEND('NOENDS')
  CALL XLGAXS(XMN,xcycle,XLENIN,XLBL,NXCHR,0.,0.)
  CALL RESET('XAXEND')
  CALL RESET('XREVTX')
  CALL RESET('YREVTX')
  CALL XNONUM
  CALL YNONUM
  CALL XLGAXS(XMN,xcycle,XLENIN,' ',1,0.,YLENIN)
  CALL YGRAXS(YMN,ystep,YMX,YLENIN,' ',1,XLENIN,0.)
ELSEIF (LOGX.EQ.1 .AND. LOGY.EQ.1) THEN  ! --loglog--
  CALL XTICKS(1)
  CALL YTICKS(1)
  CALL GRAF(XMN,XSTEP,XMX,YMN,ystep,YMX)
  IF (IXS.EQ.0.OR.XMN.LE.0.OR.XMX.LE.0) CALL LOGSCALE(XMN,XMX,DMNX,DMXX)
  XCYCLE = XLENIN / ALOG10(XMX/XMN)

```

```

IF (.NOT. POWER10 (XMN) .AND. POWER10 (XMX)) CALL XAXEND ('NOFIRST')
IF (POWER10 (XMN) .AND. .NOT. POWER10 (XMX)) CALL XAXEND ('NOLAST')
IF (.NOT. POWER10 (XMN) .AND. .NOT. POWER10 (XMX)) CALL XAXEND ('NOENDS')
CALL XREVTX
CALL RESET ('XNONUM')
CALL XLGAXS (XMN, XCYLE, XLENIN, XLBL, NXCHR, 0., 0.)
CALL RESET ('XAXEND')
CALL RESET ('XREVTX')
CALL XNONUM
CALL XLGAXS (XMN, XCYLE, XLENIN, ' ', 1, 0., YLENIN)
IF (IYS.EQ.0.OR.YMN.LE.0.OR.YMX.LE.0) CALL LOGSCALE (YMN, YMX, DMNY, DMXY)
YCYLE = YLENIN / ALOG10 (YMX/YMN)
IF (.NOT. POWER10 (YMN) .AND. POWER10 (YMX)) CALL YAXEND ('NOFIRST')
IF (POWER10 (YMN) .AND. .NOT. POWER10 (YMX)) CALL YAXEND ('NOLAST')
IF (.NOT. POWER10 (YMN) .AND. .NOT. POWER10 (YMX)) CALL YAXEND ('NOENDS')
CALL YREVTX
CALL RESET ('YNONUM')
CALL YLGAXS (YMN, YCYLE, YLENIN, YLBL, NYCHR, 0., 0.)
CALL RESET ('YAXEND')
CALL RESET ('YREVTX')
CALL YNONUM
CALL YLGAXS (YMN, YCYLE, YLENIN, ' ', 1, XLENIN, 0.)
ENDIF
CALL XREVTX
CALL YREVTX
CALL RESET ('XNONUM')
CALL RESET ('YNONUM')
C CALL BLCURV (.06, .06)
CALL THKFRM (.03)
CALL FRAME
CALL SCLPIC (SIZSYM/SIZLAB)
XMININ=-XPOSN ('LEFT', 'LOWER')
YMININ=-YPOSN ('LEFT', 'LOWER')
NPKNT=0
DO 82 M=1, NCURV
  IBEG=NPKNT+1
  NPKNT=NPKNT+NP (M)
  IF (LSKIP (M) .EQ. 0) LSKIP (M)=MAX (1, NP (M) / 25)
  CALL MARKER (LSYMBOL (M))
  IF (LBOLD (M) .EQ. 1) CALL THKCRV (.03)
  CALL NEWCLR ('FORE')
  IF (LCOLOR (M) .EQ. 1) CALL NEWCLR ('RED')
  IF (LCOLOR (M) .EQ. 2) CALL NEWCLR ('GREEN')
  IF (LCOLOR (M) .EQ. 3) CALL NEWCLR ('BLUE')
  IF (LCOLOR (M) .EQ. 4) CALL NEWCLR ('MAGENTA')
  IF (LCOLOR (M) .EQ. 5) CALL NEWCLR ('YELLOW')
  IF (LCOLOR (M) .EQ. 6) CALL NEWCLR ('CYAN')
  IF (LCOLOR (M) .EQ. 7) CALL NEWCLR ('WHITE')
  CALL HWSAT (1.0)
  CALL HWINT (1.0)
  IF (LTYPE (M) .EQ. -2) CALL CURVE (X (IBEG), Y (IBEG), NP (M), -LSKIP (M))
  IF (LTYPE (M) .EQ. -1) CALL CURVE (X (IBEG), Y (IBEG), NP (M), LSKIP (M))
  IF (LTYPE (M) .EQ. 1) CALL DASH
  IF (LTYPE (M) .EQ. 2) CALL DOT
  IF (LTYPE (M) .EQ. 3) CALL CHNDSH
  IF (LTYPE (M) .EQ. 4) CALL CHNDOT
  IF (LTYPE (M) .GT. -1 .AND. LTYPE (M) .LT. 5) THEN
    CALL LEGLIN

```

```

      CALL CURVE(X(IBEG),Y(IBEG),NP(M),0)
    ENDIF
    IF (NUMCHR(LEGTXT(M)).EQ.0) CALL DELLEG(M)
    CALL RESET('DASH')
    CALL RESET('DOT')
    CALL RESET('CHNDSH')
    CALL RESET('CHNDOT')
    CALL RESET('THKCRV')
    CALL RESET('LEGLIN')
82  CONTINUE
    CALL NEWCLR('FORE')
    IF (NSTYLNS.GT.0) THEN ! -- if story exists... --
      CALL LSTORY(IPAKSTY,NSTYLNS,XSTYPOS,YSTYPOS)
    ENDIF
    IF (NMESLNS.GT.0) THEN
      DO 83 J=1,NMESLNS
        XMESPOS=XMESF(J)*XLENIN
        YMESPOS=YMESF(J)*YLENIN
        CALL HEIGHT(SIZMES)
        CALL ANGLE(ANGMES(J))
        NMES=NUMCHR(MESTXT(J))
        CALL MESSAG(MESTXT(J),NMES,XMESPOS,YMESPOS)
83  CONTINUE
      ENDIF
      IF (LEG.EQ.1) THEN ! -- if legend exists... --
        CALL HEIGHT(SIZLEG)
        CALL LEGEND(IPAKLEG,NCURV,XLEGPOS,YLEGPOS)
      ENDIF
      CALL ENDPL(0)
      CALL DONEPL
      DO 52 J=1,NPT
        IF (XMULT.NE.0.) X(J)=X(J)/XMULT
52  IF (YMULT.NE.0.) Y(J)=Y(J)/YMULT
        IF (TERMTYPE.EQ.'REG'.OR.TERMTYPE.EQ.'340'.OR.TERMTYPE.EQ.'240') THEN
          WRITE(*,(''$',80A1)) 27,91,50,74 ! (ESC)[2J *** clear screen
          ***
          WRITE(*,(''$',80A1)) 27,91,48,59,48,72 ! (ESC)[0;0H *** home
          ***
        ELSE
C  ----- switch kermit back to text mode -----
          WRITE(*,(''$',80A1)) 27,91,63,51,56,108 ! (ESC)[?38l
        ENDIF
61  CONTINUE
        WRITE(*,(''$Plot Code [Redraw]: '''))
        IF (NCHRR(ALINE).EQ.0) GO TO 1
        LET=ALINE(1:NUMCHR(ALINE)-1)
        DO 53 I=1,NUMCHR(LET)
          ICHR = ICHAR(LET(I:I))
53  IF (ICHR.GT.96.AND.ICHR.LT.123) LET(I:I)=CHAR(ICHR-32)
          IF (LET.EQ.'E'.OR.LET.EQ.'Q') GO TO 63
          IF (LET.EQ.'S') THEN
            NC=NUMCHR(FILE)
            LET='M'
            WRITE(*,('$(A)scii or (M)etafile? [M]: '''))
            IF (NCHRR(ALINE).GT.0) LET=ALINE(1:1)
            IF (LET.EQ.'A'.OR.LET.EQ.'a') THEN
              IF (NC.EQ.0) THEN
                OUTFILE='DATA.PLT'

```

```

ELSE
  OUTFILE=FILE(1:NC-3) //'PLT'
ENDIF
WRITE(*, ' (' Ascii Filename? [' ',A,' ']/18(' ' '),$)') OUTFILE
IF (NCHRR(ALINE).GT.0) OUTFILE=ALINE(1:NUMCHR(ALINE)-1)
OPEN (11,FILE=OUTFILE,STATUS='NEW',CARRIAGECONTROL='LIST')
CALL SAVEASC(NCURV,X,Y,NP)
CLOSE(11)
MODIFIED=.FALSE.
ELSE
  IF (NC.EQ.0) THEN
    FILE='DATA.MET'
  ELSE
    FILE(NC-3:NC) = '.MET'
  ENDIF
  CALL COMPRS
  CALL HWSHD
  CALL HWCHAR(1.,1.,1.,1,'STANDARD')
  WRITE(*, ' (' Metafile Name? [' ',A,' ']/17(' ' '),$)') FILE
  IF (NCHRR(ALINE).GT.0) FILE=ALINE(1:NUMCHR(ALINE)-1)
  CALL POPNAM(FILE,NUMCHR(FILE))
  DISPLAY=.FALSE.
  GO TO 2
ENDIF
GO TO 61
ENDIF
IF (LET.EQ.'H') THEN
  PRINT *, 'Enter one of the following codes:'
  PRINT *, 'AM create/modify annotation messages'
  PRINT *, 'AS create/modify annotation story'
  PRINT *, 'E (or Q) exit'
  PRINT *, 'F cycle through fonts SIMPLX, TRIPLX, SWISSL, and SERIF'
  PRINT *, 'G toggle grid lines'
  PRINT *, 'H help'
  PRINT *, 'LB modify line LBOLD'
  PRINT *, 'LC modify line LCOLOR'
  PRINT *, 'LE create/modify legend'
  PRINT *, 'LS modify line symbols'
  PRINT *, 'LT modify line LTYPE'
  PRINT *, 'LX toggle linear/log x-axis'
  PRINT *, 'LY toggle linear/log y-axis'
  PRINT *, 'PS modify plot size or position'
  PRINT *, 'S save plot in a metafile'
  PRINT *, 'SH toggle shading of SWISSL and SERIF fonts'
  PRINT *, 'T modify plot title'
  PRINT *, 'XL modify abscissa label'
  PRINT *, 'XS modify x min, max(autoscale if = min), step, multiplier'
  PRINT *, 'YL modify ordinate label'
  PRINT *, 'YS modify y min, max(autoscale if = min), step, multiplier'
  PRINT *
  GO TO 61
ENDIF
MODIFIED=.TRUE.
IF (LET.EQ.'F') IFONT=MOD(IFONT+1,4)
IF (LET.EQ.'SH') ISHAD=IABS(ISHAD-1)
IF (LET.EQ.'LB') THEN
  WRITE(*, ' ($Line Bold [' ',I2,' ']: ' ')) (LBOLD(J),J=1,NCURV)
  IF (NCHRR(ALINE).GT.0) READ(ALINE,*) (LBOLD(J),J=1,NCURV)

```

```

ENDIF
IF (LET.EQ.'LC') THEN
  WRITE(*,'('$Line Color ['',I2,'']: '')') (LCOLOR(J),J=1,NCURV)
  IF (NCHRR(ALINE).GT.0) READ(ALINE,*) (LCOLOR(J),J=1,NCURV)
ENDIF
IF (LET.EQ.'LS') THEN
  WRITE(*,'('$Symbol # ['',,,']: '')') (LSYMBOL(J),J=1,NCURV)
  IF (NCHRR(ALINE).GT.0) READ(ALINE,*) (LSYMBOL(J),J=1,NCURV)
  WRITE(*,'('$Symbol Skip ['',I4,'']: '')') (LSKIP(J),J=1,NCURV)
  IF (NCHRR(ALINE).GT.0) READ(ALINE,*) (LSKIP(J),J=1,NCURV)
  WRITE(*,'('$Symbol Size ['',F4.2,'']: '')') SIZSYM
  IF (NCHRR(ALINE).GT.0) READ(ALINE,*) SIZSYM
ENDIF
IF (LET.EQ.'LT') THEN
  WRITE(*,'('$Line Type ['',I3,'']: '')') (LTYPE(J),J=1,NCURV)
  IF (NCHRR(ALINE).GT.0) READ(ALINE,*) (LTYPE(J),J=1,NCURV)
ENDIF
IF (LET.EQ.'XL') THEN
  WRITE(*,'('$ X Label ['',A,'']''/10('' ''),$)') XLBL
  IF (NCHRR(ALINE).GT.0) XLBL=ALINE(1:NUMCHR(ALINE)-1)
  WRITE(*,'('$Chrsiz ['',F4.2,'']: '')') SIZLAB
  IF (NCHRR(ALINE).GT.0) READ(ALINE,*) SIZLAB
ENDIF
IF (LET.EQ.'XS') THEN
  XMULTOLD=XMULT
  WRITE(*,'('$ Xmin, Xmax, Xstep, Xmultiplier ['',1P4E10.2,'']: '')')
:   XMN,XX, XSTEP, XMULT
  IF (NCHRR(ALINE).GT.0) READ(ALINE,*) XMN,XX, XSTEP, XMULT
  IF (XMULT.NE.XMULTOLD.AND.NXCHR.GT.1) XLBL=XLBL(1:NUMCHR(XLBL))/' ' ?
  XMULTOLD=XMULT
  IXS=1
  IF (XMN.EQ.XX) IXS=0
ENDIF
IF (LET.EQ.'LX') THEN
  CALL SCALE(X,NPT,XMN,XSTEP,XX,DMNX,DMXX)
  IF (DMNX.LE.0.0) THEN
    PRINT *,'Xmin must be 0.0 ... LX cancelled ... '
  ELSE
    LOGX=IABS(LOGX-1)
  ENDIF
ENDIF
IF (LET.EQ.'YL') THEN
  WRITE(*,'('$ Y Label ['',A,'']''/10('' ''),$)') YLBL
  IF (NCHRR(ALINE).GT.0) YLBL=ALINE(1:NUMCHR(ALINE)-1)
  WRITE(*,'('$Chrsiz ['',F4.2,'']: '')') SIZLAB
  IF (NCHRR(ALINE).GT.0) READ(ALINE,*) SIZLAB
ENDIF
IF (LET.EQ.'YS') THEN
  YMULTOLD=YMULT
  WRITE(*,'('$ Ymin, Ymax, Ystep, Ymultiplier ['',1P4E10.2,'']: '')')
:   YMN,YX, YSTEP, YMULT
  IF (NCHRR(ALINE).GT.0) READ(ALINE,*) YMN,YX, YSTEP, YMULT
  IF (YMULT.NE.YMULTOLD.AND.NYCHR.GT.1) YLBL=YLBL(1:NUMCHR(YLBL))/' ' ?
  YMULTOLD=YMULT
  IYS=1
  IF (YMN.EQ.YX) IYS=0
ENDIF
IF (LET.EQ.'LY') THEN

```

```

CALL SCALE(Y,NPT, YMN, YSTEP, YMX, DMNY, DMXY)
IF (DMNY.LE.0.0) THEN
  PRINT *, 'Ymin must be 0.0 ... LY cancelled ... '
ELSE
  LOGY=IABS(LOGY-1)
ENDIF
ENDIF
IF (LET.EQ.'T') THEN
  WRITE(*, ' (' Title [' ,A,' ]'/8(' ' '), $) ' ) TITLE
  IF (NCHRR(ALINE).GT.0) TITLE=ALINE(1:NUMCHR(ALINE)-1)
  WRITE(*, ' (' $Chrsiz [' ,F4.2,' ]: ' ) ' ) SIZTIT
  IF (NCHRR(ALINE).GT.0) READ(ALINE,*) SIZTIT
  WRITE(*, ' (' $Title Fractional XPOS,YPOS [' ,F4.2,F6.2,' ]: ' ) ' )
: XTITF,YTITF
  IF (NCHRR(ALINE).GT.0) READ(ALINE,*) XTITF,YTITF
ENDIF
IF (LET.EQ.'LE') THEN
  WRITE(*, ' (' Legend Title [' ,A,' ]'/15(' ' '), $) ' ) LEGTIT
  IF (NCHRR(ALINE).GT.0) LEGTIT=ALINE(1:NUMCHR(ALINE)-1)
  DO 71 J=1,NCURV
  WRITE(*, ' (' Line ' ,I2,' [' ,A,' ]'/10(' ' '), $) ' ) J,LEGTXT(J)
  IF (NCHRR(ALINE).GT.0) LEGTXT(J)=ALINE(1:NUMCHR(ALINE)-1)
71 CONTINUE
  WRITE(*, ' (' $Chrsiz [' ,F4.2,' ]: ' ) ' ) SIZLEG
  IF (NCHRR(ALINE).GT.0) READ(ALINE,*) SIZLEG
  WRITE(*, ' (' $Legend Fractional XPOS,YPOS [' ,F4.2,F6.2,' ]: ' ) ' )
: XLEGF,YLEGF
  IF (NCHRR(ALINE).GT.0) READ(ALINE,*) XLEGF,YLEGF
ENDIF
IF (LET.EQ.'AM') THEN
  WRITE(*, ' (' $# Lines [' ,I2,' ]: ' ) ' ) NMESLNS
  IF (NCHRR(ALINE).GT.0) READ(ALINE,*) NMESLNS
  IF (NMESLNS.GT.50) NMESLNS=50
  IF (NMESLNS.GT.0) THEN
    DO 12 J=1,NMESLNS
    WRITE(*, ' (' Line ' ,I2,' [' ,A,' ]'/10(' ' '), $) ' ) J,MESTXT(J)
    IF (NCHRR(ALINE).GT.0) MESTXT(J)=ALINE(1:NUMCHR(ALINE)-1)
    WRITE(*, ' (' $Fractional XPOS,YPOS,ANG [' ,2F6.2,
: F6.1,' ]: ' ) ' ) XMESF(J),YMESF(J),ANGMES(J)
    IF (NCHRR(ALINE).GT.0) READ(ALINE,*) XMESF(J),YMESF(J),ANGMES(J)
12 CONTINUE
    WRITE(*, ' (' $Chrsiz [' ,F4.2,' ]: ' ) ' ) SIZMES
    IF (NCHRR(ALINE).GT.0) READ(ALINE,*) SIZMES
  ENDIF
ENDIF
IF (LET.EQ.'AS') THEN
  WRITE(*, ' (' $# Lines [' ,I2,' ]: ' ) ' ) NSTYLNS
  IF (NCHRR(ALINE).GT.0) READ(ALINE,*) NSTYLNS
  IF (NSTYLNS.GT.0) THEN
    DO 14 J=1,NSTYLNS
    WRITE(*, ' (' Line ' ,I2,' [' ,A,' ]'/10(' ' '), $) ' ) J,STYTXT(J)
    IF (NCHRR(ALINE).GT.0) STYTXT(J)=ALINE(1:NUMCHR(ALINE)-1)
14 CONTINUE
    WRITE(*, ' (' $Chrsiz [' ,F4.2,' ]: ' ) ' ) SIZSTY
    IF (NCHRR(ALINE).GT.0) READ(ALINE,*) SIZSTY
    WRITE(*, ' (' $Annotation Fractional XPOS,YPOS [' ,2F6.2,' ]: ' ) ' )
: XSTYF,YSTYF
    IF (NCHRR(ALINE).GT.0) READ(ALINE,*) XSTYF,YSTYF
  ENDIF

```

```

        ENDIF
    ENDIF
    IF (LET.EQ.'PS') THEN
        WRITE(*,(' Xorigin, Yorigin, Xlength, Ylength ['',4F6.3,'']: '''))
    :   XORIGIN,YORIGIN,XLENIN,YLENIN
        IF (NCHRR(ALINE).GT.0) READ(ALINE,*) XORIGIN,YORIGIN,XLENIN,YLENIN
    ENDIF
    GO TO 61
63 CONTINUE
    RETURN
    END

    LOGICAL FUNCTION POWER10 (ANUM)
    A=ALOG10 (ANUM)
    B=IFIX (A)
    POWER10 = (ABS (A-B) .LT.1.E-6)
    RETURN
    END

    CHARACTER*3 FUNCTION VMSTERM()
    INCLUDE ' ($DVIDEF) '
    INTEGER LIB$GETDVI
    CHARACTER*5 VT2XX,REGIS,VT3XX
* default to tektronics terminal type
    VMSTERM='TEK'
* VT240 overrides tektronics
    ISTATUS=LIB$GETDVI (DVI$_TT_DECCRT2,, 'TT',,VT2XX,NCHAR)
    IF (VT2XX.EQ.'TRUE') VMSTERM='240'
* regis overrides VT240
    ISTATUS=LIB$GETDVI (DVI$_TT_REGIS,, 'TT',,REGIS,NCHAR)
    IF (REGIS.EQ.'TRUE') VMSTERM='REG'
* VT340 overrides regis
    ISTATUS=LIB$GETDVI (DVI$_TT_DECCRT3,, 'TT',,VT3XX,NCHAR)
    IF (VT3XX.EQ.'TRUE') VMSTERM='340'
    RETURN
    END

    SUBROUTINE SAVEASC (NCURV,X,Y,NP)
    CHARACTER*60 TITLE,XLBL,YLBL,LEGTIT*20,LEGTXT (50) ,STYTXT (50) ,MESTXT (50)
    COMMON/PLOTSTORY/NSTYLNS,SIZSTY,STYTXT,XSTYF,YSTYF
    COMMON/PLOTLEGEND/XLEGF,YLEGF,SIZLEG,LEGTIT,LEGTXT
    COMMON/PLOT LINES/LBOLD (50) ,LCOLOR (50) ,LTYPE (50) ,
    :   LSYMBOL (50) ,LSKIP (50) ,SIZSYM
    COMMON/PLOT LABEL/XTITF,YTITF,SIZTIT,TITLE,SIZLAB,XLBL,YLBL,IFONT,ISHAD
    COMMON/PLOT AXES/IXS,XMN,XSTEP,XXM,XMULT,LOGX,
    :   IYS,YMN,YSTEP,YMX,YMULT,LOGY
    COMMON/PLOT SIZE/XORIGIN,YORIGIN,XLENIN,YLENIN
    COMMON/MESSAGE/NMESLNS,SIZMES,MESTXT,XMESF (50) ,YMESF (50) ,ANGMES (50)
    DIMENSION X(*),Y(*),NP(*)
    WRITE(11,(' 'plot title ' ',A')) TITLE(1:NUMCHR(TITLE))
    WRITE(11,(' 'title position ' ',2F6.3')) XTITF,YTITF
    WRITE(11,(' 'title size ' ',F6.3')) SIZTIT
    WRITE(11,(' 'x axis label ' ',A')) XLBL(1:NUMCHR(XLBL))
    WRITE(11,(' 'x axis min max step multiplier ' ',
    :   1P4E10.3')) XMN,XXM,XSTEP,XXMULT
    IF (LOGX.EQ.1) WRITE(11,(' 'x axis log ' '))
    WRITE(11,(' 'y axis label ' ',A')) YLBL(1:NUMCHR(YLBL))
    WRITE(11,(' 'y axis min max step multiplier ' ',

```

```

:                                     1P4E10.3)') YMN,YMX,YSTEP,YMULT
IF (LOGY.EQ.1) WRITE(11,('y axis log'))
WRITE(11,('axis label size ',F6.3)) SIZLAB
WRITE(11,('x-y origin x-y length ',4F6.3))
: XORIGIN,YORIGIN,XLENIN,YLENIN
IF (IFONT.EQ.1) WRITE(11,('font triplx'))
IF (IFONT.EQ.2.AND.ISHAD.EQ.0) WRITE(11,('font swissl'))
IF (IFONT.EQ.2.AND.ISHAD.EQ.1) WRITE(11,('font swissl shaded'))
IF (IFONT.EQ.3.AND.ISHAD.EQ.0) WRITE(11,('font serif'))
IF (IFONT.EQ.3.AND.ISHAD.EQ.1) WRITE(11,('font serif shaded'))
WRITE(11,('legend title ',A)) LEGTIT(1:NUMCHR(LEGTIT))
WRITE(11,('legend position ',2F6.3)) XLEGF,YLEGF
WRITE(11,('legend size ',F6.3)) SIZELEG
DO 12 I=1,NMESLNS
WRITE(11,('message entry ',A)) MESTXT(I)(1:NUMCHR(MESTXT(I)))
12 WRITE(11,('message position, angle ',2F6.3,F6.1))
:                                     XMESF(I),YMESF(I),ANGMES(I)
WRITE(11,('message size ',F6.3)) SIZMES
DO 10 I=1,NSTYLNS
10 WRITE(11,('story entry ',A)) STYTXT(I)(1:NUMCHR(STYTXT(I)))
WRITE(11,('story position ',2F6.3)) XSTYF,YSTYF
WRITE(11,('story size ',F6.3)) SIZSTY
WRITE(11,('symbol size ',F6.3)) SIZSYM
K = 0
DO 11 I=1,NCURV
WRITE(11,('new curve'))
IF (NUMCHR(LEGTXT(I)).GT.0) WRITE(11,('legend entry ',A))
: LEGTXT(I)(1:NUMCHR(LEGTXT(I)))
IF (LCOLOR(I).EQ.1) WRITE(11,('line color red'))
IF (LCOLOR(I).EQ.2) WRITE(11,('line color green'))
IF (LCOLOR(I).EQ.3) WRITE(11,('line color blue'))
IF (LCOLOR(I).EQ.4) WRITE(11,('line color magenta'))
IF (LCOLOR(I).EQ.5) WRITE(11,('line color yellow'))
IF (LCOLOR(I).EQ.6) WRITE(11,('line color cyan'))
IF (LCOLOR(I).EQ.7) WRITE(11,('line color white'))
IF (LBOLD(I).EQ.1) WRITE(11,('line type bold'))
IF (LTYPE(I).EQ.-2) WRITE(11,('line type symbols only'))
IF (LTYPE(I).EQ.-1) WRITE(11,('line type connected symbols'))
IF (LTYPE(I).EQ.1) WRITE(11,('line type dashed'))
IF (LTYPE(I).EQ.2) WRITE(11,('line type dotted'))
IF (LTYPE(I).EQ.3) WRITE(11,('line type chain-dashed'))
IF (LTYPE(I).EQ.4) WRITE(11,('line type chain-dotted'))
IF (LTYPE(I).LT.0) THEN
WRITE(11,('symbol skip ',I1)) LSKIP(I)
IF (LSYMBOL(I).GT.0) WRITE(11,('symbol number ',I2)) LSYMBOL(I)
ENDIF
DO 11 J=1,NP(I)
K = K+1
11 WRITE(11,('1P2E12.4')) X(K),Y(K)
RETURN
END

SUBROUTINE SCALE(DATA,NP,SMIN,SSTP,SMAX,DMIN,DMAX)
DIMENSION DATA(NP)
DMIN=DATA(1)
DMAX=DATA(1)
DO 10 J=2,NP
DMIN=MIN(DMIN,DATA(J))

```

```

10 DMAX=MAX (DMAX, DATA (J) )
   DMN=DMIN
   DMX=DMAX
   IF (DMIN.EQ.DMAX) THEN
     DMN=DMN-1.
     DMX=DMX+1.
   ENDIF
   DRNG=ABS (DMX-DMN)
   DO 11 J=1, 60
     FAC=10.** (J-31)
     IF ( (DRNG*FAC) .GE.3.9) GO TO 12
11 CONTINUE
12 CONTINUE
   SSTP=1./FAC
   CALL LINSKALE (SMIN, SMAX, DMN, DMX, SSTP, NLINES)
   IF (NLINES.LE.8) GO TO 40
   SSTP=2./FAC
   CALL LINSKALE (SMIN, SMAX, DMN, DMX, SSTP, NLINES)
   IF (NLINES.LE.8) GO TO 40
   SSTP=5./FAC
   CALL LINSKALE (SMIN, SMAX, DMN, DMX, SSTP, NLINES)
   IF (NLINES.LE.8) GO TO 40
   FAC=FAC*.1
   GO TO 12
40 CONTINUE
   RETURN
   END

   SUBROUTINE LINSKALE (SMIN, SMAX, DMN, DMX, SSTP, NLINES)
     SMIN=IFIX (DMN/SSTP+.5) *SSTP
     SMAX=IFIX (DMX/SSTP+.5) *SSTP
* allow an extra border around data
C     IF (SMIN.GT.DMN- (SMAX-SMIN) *.04) SMIN=SMIN-SSTP
C     IF (SMIN.GT.DMN- (SMAX-SMIN) *.04) SMIN=SMIN-SSTP
C     IF (SMAX.LT.DMX+ (SMAX-SMIN) *.04) SMAX=SMAX+SSTP
C     IF (SMAX.LT.DMX+ (SMAX-SMIN) *.04) SMAX=SMAX+SSTP
* allow data to touch axes
     IF (SMIN.GT.DMN) SMIN=SMIN-SSTP
     IF (SMIN.GT.DMN) SMIN=SMIN-SSTP
     IF (SMAX.LT.DMX) SMAX=SMAX+SSTP
     IF (SMAX.LT.DMX) SMAX=SMAX+SSTP
     NLINES= (SMAX-SMIN) /SSTP+.5
     RETURN
     END

   SUBROUTINE LOGSCALE (SMIN, SMAX, DATAMN, DATAMX)
     CYCLES=ALOG10 (DATAMX/DATAMN)
     FAC=CYCLES*.5+1.
     SMIN = DATAMN/FAC
     SMAX = DATAMX*FAC
     RETURN
     END

   FUNCTION NUMCHR (STRING)
     CHARACTER STRING* (*), BLNK, NUL
     PARAMETER (NUL=CHAR(0), BLNK=' ')
     MAXCHR=LEN (STRING)
11 DO 20 NUMCHR=MAXCHR, 1, -1

```

```

      IF (STRING (NUMCHR:NUMCHR) .NE. BLNK. AND. STRING (NUMCHR:NUMCHR) .NE. NUL)
      :RETURN
20 CONTINUE
      RETURN
      END

      FUNCTION NCHRR (ALINE)
      CHARACTER ALINE* (40)
      READ ' (Q,A)', NCHRR, ALINE
      IF (ALINE (1:NCHRR) .EQ. ' ') ALINE (1:NCHRR) = ' '
      IF (NCHRR.GT.0) ALINE (NCHRR+1:NCHRR+1) = '/'
      RETURN
      END

```

Appendix C

PC Version of PLOTDATA

```
*****
*          PC Version of PLOTDATA Source Code, January 1991          *
*          MAXPTS = max total points  MC = max curves                *
*****

PARAMETER (MAXPTS=3000, MC=50)
DIMENSION X(MAXPTS), Y(MAXPTS), NP(MC)
LOGICAL ERRORS,MODIFIED,EXIST
CHARACTER*80 FILE*40, OUTFILE*40, ALINE, INPUT*1, ARGV*40
INTEGER*2 ARGV, NARGPOS, NARGCHAR
COMMON/INFO/MODIFIED
OPEN (99,STATUS='SCRATCH')
NUMARGS = ARGV()
NARGPOS = 2
FILE = ARGV(NARGPOS)
IF (NUMARGS.LT.3.OR.NUMCHR(FILE).EQ.0) THEN
    FILE='DATA.PLT'
    WRITE (*,'('' Plot data filename? [DATA.PLT]: '')')
    IF (NCHRR(ALINE).GT.0) FILE=ALINE(1:NUMCHR(ALINE)-1)
ENDIF
10 INQUIRE (FILE=FILE,EXIST=EXIST)
IF (.NOT.EXIST) THEN
    WRITE (*,'('' File does not exist, reenter: '')')
    READ (*,'(A)') FILE
    GO TO 10
ENDIF
OPEN (10,FILE=FILE,STATUS='UNKNOWN')
CALL READASC (ALINE,NCURV,X,Y,NP,ERRORS,MAXPTS)
IF (.NOT.ERRORS) THEN
    CALL TPLOT (NCURV,X,Y,NP)
    IF (MODIFIED) THEN
        INPUT='Y'
        WRITE (*,'('' Save changes to an ascii file? [Y]: '')')
        IF (NCHRR(ALINE).GT.0) INPUT=ALINE(1:NUMCHR(ALINE)-1)
        IF (INPUT.EQ.'Y'.OR.INPUT.EQ.'y') THEN
            OUTFILE=FILE(1:NUMCHR(FILE)-3)///'PLT'
            WRITE (*,'('' Filename? ['',A,'']'/12('' ''))') OUTFILE
            IF (NCHRR(ALINE).GT.0) OUTFILE=ALINE(1:NUMCHR(ALINE)-1)
            OPEN (11,FILE=OUTFILE,STATUS='UNKNOWN')
            REWIND 10
1        READ (10,'(A)',END=2) ALINE
            NC=NUMCHR (ALINE)
            IF (NC.EQ.0) THEN
                WRITE (11,*)
                GO TO 1
            ELSEIF (ALINE(1:1).EQ.'*' .OR. ALINE(1:1).EQ.'!') THEN
                WRITE (11,'(A)') ALINE(1:NC)
                GO TO 1
            ENDIF
2        CLOSE (10)
        CALL SAVEASC (NCURV,X,Y,NP)
```

```

        ENDFILE 11
        CLOSE(11)
    ENDIF
ENDIF
ELSE
    PAUSE 'Press <RETURN> to continue...'
ENDIF
STOP
END

SUBROUTINE READASC(ALINE,NCURV,X,Y,NP,ERRORS,MAXPTS)
*      MC = max curves      MS = max story lines      MM = max messages
PARAMETER (MC=50, MS=20, MM=10)
LOGICAL ERRORS, XYDATA
CHARACTER ALINE*(*), CLINE*80
DIMENSION NP(*),X(*),Y(*)
CHARACTER*60 STRIMP,TITLE,XLBL,YLBL,MESTXT(MM)
CHARACTER*59 LEGTIT*20,LEGTX(MC),STYTX(MS)
COMMON/STORY/NSTYLS,SIZSTY,STYTX,XSTYF,YSTYF
COMMON/LEGEND/XLEGF,YLEGF,SIZLEG,LEGTIT,LEGTX
COMMON/LINES/LBOLD(MC),LCOLOR(MC),LTYPE(MC),
:      LSYMBOL(MC),LSKIP(MC),SIZSYM
COMMON/LABEL/XTITF,YTITF,SIZTIT,TITLE,SIZLAB,XLBL,YLBL,IFONT,ISHAD
COMMON/AXES/IXS,XMN,XSTEP,XMX,XMULT,LOGX,
:      IYS,YMN,YSTEP,YMX,YMULT,LOGY
COMMON/PLOTSIZE/XORIGIN,YORIGIN,XLENIN,YLENIN
COMMON/MESSAGE/NMESLS,SIZMES,MESTXT,XMESF(MM),YMESF(MM),
:      ANGMES(MM)
DO 4 I=1,MC
4      NP(I)=0
      NSTYLS = 0
      NMESLS = 0
      ILINE = 0
      NCURV = 1
      IP = 0
      ERRORS = .FALSE.
      XYDATA=.FALSE.
2      ILINE=ILINE+1
      READ(10,'(A)',END=3) ALINE
      NC=NUMCHR(ALINE)
      IF(NC.EQ.0.OR.ALINE(1:1).EQ.'!' .OR.ALINE(1:1).EQ.'**') GO TO 2
      CLINE = ' '
      DO 10 I=1,NC
          ICHR = ICHAR(ALINE(I:I))
          IF(ICHR.GT.96.AND.ICHR.LT.123) THEN
              CLINE(I:I)=CHAR(ICHR-32)
          ELSE
              CLINE(I:I)=CHAR(ICHR)
          ENDIF
10      CONTINUE
      IF(CLINE(1:9).EQ.'NEW CURVE') THEN
          XYDATA=.FALSE.
          IF(NP(NCURV).GT.0) NCURV = NCURV+1
          IF(NCURV.GT.MC) THEN
              PRINT *, 'Maximum number of curves exceeded.'
              ERRORS=.TRUE.
              RETURN
          ENDIF

```

```

ELSEIF (XYDATA) THEN
  GO TO 5
* plotstory *
ELSEIF (CLINE(1:14).EQ.'STORY POSITION') THEN
  CALL WRITE99 (ALINE,16,NC)
  READ (99,*,ERR=1) XSTYF,YSTYF
ELSEIF (CLINE(1:10).EQ.'STORY SIZE') THEN
  CALL WRITE99 (ALINE,12,NC)
  READ (99,*,ERR=1) SIZSTY
ELSEIF (CLINE(1:11).EQ.'STORY ENTRY') THEN
  NSTYLS = NSTYLS+1
  IF (NC.GE.13) STYTXT (NSTYLS) = ALINE(13:NC)
* message *
ELSEIF (CLINE(1:12).EQ.'MESSAGE SIZE') THEN
  CALL WRITE99 (ALINE,14,NC)
  READ (99,*,ERR=1) SIZMES
ELSEIF (CLINE(1:13).EQ.'MESSAGE ENTRY') THEN
  NMESLS = NMESLS+1
  IF (NC.GE.15) MESTXT (NMESLS) = ALINE(15:NC)
ELSEIF (CLINE(1:23).EQ.'MESSAGE POSITION, ANGLE') THEN
  IF (NMESLS.EQ.0) GO TO 1
  CALL WRITE99 (ALINE,25,NC)
  READ (99,*,ERR=1) XMESF (NMESLS), YMESF (NMESLS), ANGMS (NMESLS)
* plotlegend *
ELSEIF (CLINE(1:15).EQ.'LEGEND POSITION') THEN
  CALL WRITE99 (ALINE,17,NC)
  READ (99,*,ERR=1) XLEGF,YLEGF
ELSEIF (CLINE(1:11).EQ.'LEGEND SIZE') THEN
  CALL WRITE99 (ALINE,13,NC)
  READ (99,*,ERR=1) SIZLEG
ELSEIF (CLINE(1:12).EQ.'LEGEND TITLE') THEN
  IF (NC.GE.14) LEGTIT = ALINE(14:NC)
ELSEIF (CLINE(1:12).EQ.'LEGEND ENTRY') THEN
  IF (NC.GE.14) LEGTXT (NCURV) = ALINE(14:NC)
* plotlines *
ELSEIF (CLINE(1:14).EQ.'LINE COLOR RED') THEN
  LCOLOR (NCURV) = 1
ELSEIF (CLINE(1:16).EQ.'LINE COLOR GREEN') THEN
  LCOLOR (NCURV) = 2
ELSEIF (CLINE(1:15).EQ.'LINE COLOR BLUE') THEN
  LCOLOR (NCURV) = 3
ELSEIF (CLINE(1:18).EQ.'LINE COLOR MAGENTA') THEN
  LCOLOR (NCURV) = 4
ELSEIF (CLINE(1:17).EQ.'LINE COLOR YELLOW') THEN
  LCOLOR (NCURV) = 5
ELSEIF (CLINE(1:15).EQ.'LINE COLOR CYAN') THEN
  LCOLOR (NCURV) = 6
ELSEIF (CLINE(1:16).EQ.'LINE COLOR WHITE') THEN
  LCOLOR (NCURV) = 7
ELSEIF (CLINE(1:14).EQ.'LINE TYPE BOLD') THEN
  LBOLD (NCURV) = 1
ELSEIF (CLINE(1:22).EQ.'LINE TYPE SYMBOLS ONLY') THEN
  LTYPE (NCURV) = -2
ELSEIF (CLINE(1:27).EQ.'LINE TYPE CONNECTED SYMBOLS') THEN
  LTYPE (NCURV) = -1
ELSEIF (CLINE(1:16).EQ.'LINE TYPE DASHED') THEN
  LTYPE (NCURV) = 1
ELSEIF (CLINE(1:16).EQ.'LINE TYPE DOTTED') THEN

```

```

    LTYPE(NCURV) = 2
ELSEIF (CLINE(1:22).EQ.'LINE TYPE CHAIN-DASHED') THEN
    LTYPE(NCURV) = 3
ELSEIF (CLINE(1:22).EQ.'LINE TYPE CHAIN-DOTTED') THEN
    LTYPE(NCURV) = 4
ELSEIF (CLINE(1:13).EQ.'SYMBOL NUMBER') THEN
    CALL WRITE99(ALINE,15,NC)
    READ(99,*,ERR=1) LSYMBOL(NCURV)
ELSEIF (CLINE(1:11).EQ.'SYMBOL SKIP') THEN
    CALL WRITE99(ALINE,13,NC)
    READ(99,*,ERR=1) LSKIP(NCURV)
ELSEIF (CLINE(1:11).EQ.'SYMBOL SIZE') THEN
    CALL WRITE99(ALINE,13,NC)
    READ(99,*,ERR=1) SIZSYM
* plotlabel *
ELSEIF (CLINE(1:14).EQ.'TITLE POSITION') THEN
    CALL WRITE99(ALINE,16,NC)
    READ(99,*,ERR=1) XTITF,YTITF
ELSEIF (CLINE(1:10).EQ.'TITLE SIZE') THEN
    CALL WRITE99(ALINE,12,NC)
    READ(99,*,ERR=1) SIZTIT
ELSEIF (CLINE(1:10).EQ.'PLOT TITLE') THEN
    IF (NC.GE.12) TITLE = ALINE(12:NC)
ELSEIF (CLINE(1:15).EQ.'AXIS LABEL SIZE') THEN
    CALL WRITE99(ALINE,17,NC)
    READ(99,*,ERR=1) SIZLAB
ELSEIF (CLINE(1:12).EQ.'X AXIS LABEL') THEN
    IF (NC.GE.14) XLBL = ALINE(14:NC)
ELSEIF (CLINE(1:12).EQ.'Y AXIS LABEL') THEN
    IF (NC.GE.14) YLBL = ALINE(14:NC)
ELSEIF (CLINE(1:11).EQ.'FONT TRIPLX') THEN
    IFONT = 1
ELSEIF (CLINE(1:18).EQ.'FONT SWISSL SHADED') THEN
    IFONT = 2
    ISHAD = 1
ELSEIF (CLINE(1:17).EQ.'FONT SERIF SHADED') THEN
    IFONT = 3
    ISHAD = 1
ELSEIF (CLINE(1:11).EQ.'FONT SWISSL') THEN
    IFONT = 2
ELSEIF (CLINE(1:10).EQ.'FONT SERIF') THEN
    IFONT = 3
* plotaxes *
ELSEIF (CLINE(1:30).EQ.'X AXIS MIN MAX STEP MULTIPLIER') THEN
    IXS = 1
    CALL WRITE99(ALINE,32,NC)
    READ(99,*,ERR=1) XMN,XXM,XSTEP,XXMULT
ELSEIF (CLINE(1:10).EQ.'X AXIS LOG') THEN
    LOGX = 1
ELSEIF (CLINE(1:30).EQ.'Y AXIS MIN MAX STEP MULTIPLIER') THEN
    IYS = 1
    CALL WRITE99(ALINE,32,NC)
    READ(99,*,ERR=1) YMN,YMX,YSTEP,YMULT
ELSEIF (CLINE(1:10).EQ.'Y AXIS LOG') THEN
    LOGY = 1
* plotsize *
ELSEIF (CLINE(1:21).EQ.'X-Y ORIGIN X-Y LENGTH') THEN
    CALL WRITE99(ALINE,23,NC)

```

```

      READ (99,*,ERR=1) XORIGIN,YORIGIN,XLENIN,YLENIN
* data point *
      ELSE
5      IP=IP+1
      IF (IP.GT.MAXPTS) THEN
        WRITE (*,
: ' ('' Maximum number of points ('',I4,'') exceeded.'')' ) MAXPTS
        ERRORS=.TRUE.
        RETURN
      ENDIF
      CALL WRITE99 (ALINE,1,NC)
      READ (99,*,ERR=1) X(IP),Y(IP)
      NP (NCURV)=NP (NCURV)+1
      XYDATA=.TRUE.
      ENDIF
      GO TO 2
3      CONTINUE
      RETURN
1      WRITE (*,' ('' Error in line '',I3,'': '',80A)')
: ILINE,ALINE (1:NC)
      ERRORS = .TRUE.
      RETURN
      END

      SUBROUTINE SAVEASC (NCURV,X,Y,NP)
* MC = max curves   MS = max story lines   MM = max messages
      PARAMETER (MC=50, MS=20, MM=10)
      CHARACTER*60 STRTMP,TITLE,XLBL,YLBL,MESTXT (MM)
      CHARACTER*59 LEGTIT*20,LEGTXT (MC),STYTXT (MS)
      COMMON/STORY/NSTYLNS,SIZSTY,STYTXT,XSTYF,YSTYF
      COMMON/LEGEND/XLEGF,YLEGF,SIZLEG,LEGTIT,LEGTXT
      COMMON/LINES/LBOLD (MC),LCOLOR (MC),LTYPE (MC),
:
:                                LSYMBOL (MC),LSKIP (MC),SIZSYM
      COMMON/LABEL/XTITF,YTITF,SIZTIT,TITLE,SIZLAB,XLBL,YLBL,IFONT,ISHAD
      COMMON/AXES/IXS,XMN,XSTEP,XMX,XMULT,LOGX,
:
:                                IYS,YMN,YSTEP,YMX,YMULT,LOGY
      COMMON/PLOTSIZE/XORIGIN,YORIGIN,XLENIN,YLENIN
      COMMON/MESSAGE/NMESLNS,SIZMES,MESTXT,XMESF (MM),YMESF (MM),
:                                ANGMES (MM)
      DIMENSION X(*),Y(*),NP (*)
      WRITE (11,' (''plot title '',A)') TITLE (1:NUMCHR (TITLE))
      WRITE (11,' (''title position '',2F6.3)') XTITF,YTITF
      WRITE (11,' (''title size '',F6.3)') SIZTIT
      WRITE (11,' (''x axis label '',A)') XLBL (1:NUMCHR (XLBL))
      WRITE (11,' (''x axis min max step multiplier '',
:                                1P4E10.3)') XMN,XMX,XSTEP,XMULT
      IF (LOGX.EQ.1) WRITE (11,' (''x axis log''))
      WRITE (11,' (''y axis label '',A)') YLBL (1:NUMCHR (YLBL))
      WRITE (11,' (''y axis min max step multiplier '',
:                                1P4E10.3)') YMN,YMX,YSTEP,YMULT
      IF (LOGY.EQ.1) WRITE (11,' (''y axis log''))
      WRITE (11,' (''axis label size '',F6.3)') SIZLAB
      WRITE (11,' (''x-y origin x-y length '',4F6.3)')
: XORIGIN,YORIGIN,XLENIN,YLENIN
      IF (IFONT.EQ.1) WRITE (11,' (''font triplx''))
      IF (IFONT.EQ.2.AND.ISHAD.EQ.0) WRITE (11,' (''font swissl''))
      IF (IFONT.EQ.2.AND.ISHAD.EQ.1)
: WRITE (11,' (''font swissl shaded''))

```

```

IF (IFONT.EQ.3.AND.ISHAD.EQ.0) WRITE(11,('font serif'))
IF (IFONT.EQ.3.AND.ISHAD.EQ.1) WRITE(11,('font serif shaded'))
WRITE(11,('legend title ',A)) LEGTIT(1:NUMCHR(LEGTIT))
WRITE(11,('legend position ',2F6.3)) XLEGF,YLEGF
WRITE(11,('legend size ',F6.3)) SIZELEG
DO 12 I=1,NMESLNS
WRITE(11,('message entry ',A)) MESTXT(I)(1:NUMCHR(MESTXT(I)))
12 WRITE(11,('message position, angle ',2F6.3,F6.1))
: XMESF(I),YMESF(I),ANGMES(I)
WRITE(11,('message size ',F6.3)) SIZEMES
DO 10 I=1,NSTYLS
10 WRITE(11,('story entry ',A)) STYTXT(I)(1:NUMCHR(STYTXT(I)))
WRITE(11,('story position ',2F6.3)) XSTYF,YSTYF
WRITE(11,('story size ',F6.3)) SIZESTY
WRITE(11,('symbol size ',F6.3)) SIZESYM
K=0
DO 11 I=1,NCURV
WRITE(11,('new curve'))
WRITE(11,('legend entry ',A)) LEGTXT(I)(1:NUMCHR(LEGTXT(I)))
IF (LCOLOR(I).EQ.1) WRITE(11,('line color red'))
IF (LCOLOR(I).EQ.2) WRITE(11,('line color green'))
IF (LCOLOR(I).EQ.3) WRITE(11,('line color blue'))
IF (LCOLOR(I).EQ.4) WRITE(11,('line color magenta'))
IF (LCOLOR(I).EQ.5) WRITE(11,('line color yellow'))
IF (LCOLOR(I).EQ.6) WRITE(11,('line color cyan'))
IF (LCOLOR(I).EQ.7) WRITE(11,('line color white'))
IF (LBOLD(I).EQ.1) WRITE(11,('line type bold'))
IF (LTYPE(I).EQ.-2) WRITE(11,('line type symbols only'))
IF (LTYPE(I).EQ.-1) WRITE(11,('line type connected symbols'))
IF (LTYPE(I).EQ.1) WRITE(11,('line type dashed'))
IF (LTYPE(I).EQ.2) WRITE(11,('line type dotted'))
IF (LTYPE(I).EQ.3) WRITE(11,('line type chain-dashed'))
IF (LTYPE(I).EQ.4) WRITE(11,('line type chain-dotted'))
IF (LTYPE(I).LT.0) THEN
WRITE(11,('symbol skip ',I1)) LSKIP(I)
IF (LSYMBOL(I).GT.0) WRITE(11,('symbol number ',I2))
: LSYMBOL(I)
ENDIF
DO 11 J=1,NP(I)
K=K+1
11 WRITE(11,('1P2E12.4')) X(K),Y(K)
RETURN
END

SUBROUTINE WRITE99 (STRING,IBEGIN,IEND)
CHARACTER*(*) STRING
REWIND 99
WRITE (99, ' (A) ') STRING (IBEGIN:IEND)
REWIND 99
RETURN
END

```

Appendix D

VAX Version of PLOTDATA

```
*****
*      VAX FORTRAN Version of PLOTDATA Source Code as of January 1991      *
*      MAXPTS = max total points                                           *
*****

PARAMETER (MAXPTS=5000)
DIMENSION X(MAXPTS), Y(MAXPTS), NP(50)
LOGICAL ERRORS, MODIFIED, EXIST, COMLINE(80)
CHARACTER*80 FILE*40, OUTFILE*40, ALINE, INPUT*1, TERMINAL*3, COMPARM
EQUIVALENCE (COMLINE, COMPARM)
COMMON/INFO/MODIFIED, FILE
CALL GETMCR(COMLINE, NPARM)
IF (NPARM.GT.1) THEN
    FILE=COMPARM(2:NPARM) //' .PLT'
ELSE
    FILE='DATA.PLT'
    WRITE (*, ' (' '$Plot data (.PLT) filename? [DATA]: ' ') ' )
    IF (NCHRR(ALINE).GT.0) FILE=ALINE(1:NUMCHR(ALINE)-1) //' .PLT'
ENDIF
10 INQUIRE (FILE=FILE, EXIST=EXIST)
IF (.NOT.EXIST) THEN
    N=NUMCHR(FILE)
    WRITE (*, ' (' '$File. ' ', A, ' ' ' does not exist, reenter: ' ') ' ) FILE(1:N)
    READ (*, ' (A) ' ) FILE
    FILE=FILE(1:NUMCHR(FILE)) //' .PLT'
    GO TO 10
ENDIF
OPEN (10, FILE=FILE, STATUS='UNKNOWN')
CALL READASC (ALINE, NCURV, X, Y, NP, ERRORS, MAXPTS)
IF (.NOT.ERRORS) THEN
    CALL TPLOT (NCURV, X, Y, NP, '?')
    IF (MODIFIED) THEN
        INPUT='Y'
        WRITE (*, ' (' '$Save changes to an ascii file? [Y]: ' ') ' )
        IF (NCHRR(ALINE).GT.0) INPUT=ALINE(1:NUMCHR(ALINE)-1)
        IF (INPUT.EQ.'Y'.OR.INPUT.EQ.'y') THEN
            OUTFILE=FILE(1:NUMCHR(FILE)-3) //' PLT'
            WRITE (*, ' (' ' Filename? [ ' ', A, ' ' ' ] ' ' /12(' ' ' ' ), $) ' ) OUTFILE
            IF (NCHRR(ALINE).GT.0) OUTFILE=ALINE(1:NUMCHR(ALINE)-1)
            OPEN (11, FILE=OUTFILE, STATUS='NEW', CARRIAGECONTROL='LIST')
            REWIND 10
1        READ(10, ' (A) ', END=2) ALINE
            NC=NUMCHR(ALINE)
            IF (NC.EQ.0) THEN
                WRITE(11, *)
                GO TO 1
            ELSEIF (ALINE(1:1).EQ.'*' .OR. ALINE(1:1).EQ.'!') THEN
                WRITE(11, ' (A) ' ) ALINE(1:NC)
                GO TO 1
            ENDIF
2        CLOSE (10)
```

```

        CALL SAVEASC (NCURV,X,Y,NP)
    ENDIF
ENDIF
STOP
END

SUBROUTINE READASC (ALINE,NCURV,X,Y,NP,ERRORS,MAXPTS)
LOGICAL ERRORS, XYDATA
CHARACTER*80 ALINE*(*), CLINE
DIMENSION NP (*),X(*),Y(*)
CHARACTER*60 TITLE,XLBL,YLBL,LEGTIT*20,LEGTXT(50),STYTXT(50),MESTXT(50)
COMMON/PLOTSTORY/NSTYLS,SIZSTY,STYTXT,XSTYF,YSTYF
COMMON/PLOTLEGEND/XLEGF,YLEGF,SIZLEG,LEGTIT,LEGTXT
COMMON/PLOTLINES/LBOLD(50),LCOLOR(50),LTYPE(50),
: LSYMBOL(50),LSKIP(50),SIZSYM
COMMON/PLOTLABEL/XTITF,YTITF,SIZTIT,TITLE,SIZLAB,XLBL,YLBL,IFONT,ISHAD
COMMON/PLOTAXES/IXS,XMN,XSTEP,XMX,XMULT,LOGX,
: IYS,YMN,YSTEP,YMX,YMULT,LOGY
COMMON/PLOTSIZE/XORIGIN,YORIGIN,XLENIN,YLENIN
COMMON/MESSAGE/NMESLNS,SIZMES,MESTXT,XMESF(50),YMESF(50),ANGMES(50)
DO 4 I=1,50
4  NP(I)=0
   NSTYLS = 0
   NMESLNS = 0
   ILINE = 0
   NCURV = 1
   IP = 0
   ERRORS = .FALSE.
   XYDATA = .FALSE.
2  ILINE=ILINE+1
   READ(10,'(Q,A)',END=3) NC,ALINE
   IF(NC.EQ.0.OR.ALINE(1:1).EQ.'!' .OR.ALINE(1:1).EQ.'*') GO TO 2
   CLINE = ' '
   DO 10 I=1,NC
   ICHR = ICHAR(ALINE(I:I))
   IF(ICHR.GT.96.AND.ICHR.LT.123) THEN
       CLINE(I:I)=CHAR(ICHR-32)
   ELSE
       CLINE(I:I)=ALINE(I:I)
   ENDIF
10 CONTINUE
   IF(CLINE(1:9).EQ.'NEW CURVE') THEN
       XYDATA=.FALSE.
       IF(NP(NCURV).GT.0) NCURV = NCURV+1
       IF(NCURV.GT.50) THEN
           PRINT *,'Maximum number of curves (50) exceeded.'
           ERRORS=.TRUE.
           RETURN
       ENDIF
       ELSEIF(XYDATA) THEN
           GO TO 5
* plotstory *
       ELSEIF(CLINE(1:14).EQ.'STORY POSITION') THEN
           READ(ALINE(16:NC),*,ERR=1) XSTYF,YSTYF
       ELSEIF(CLINE(1:10).EQ.'STORY SIZE') THEN
           READ(ALINE(12:NC),*,ERR=1) SIZSTY
       ELSEIF(CLINE(1:11).EQ.'STORY ENTRY') THEN

```

```

      NSTYLNS = NSTYLNS+1
      IF (NC.GE.13) STYTXT(NSTYLNS) = ALINE(13:NC)
* message *
      ELSEIF (CLINE(1:12).EQ.'MESSAGE SIZE') THEN
        READ (ALINE(14:NC),*,ERR=1) SIZMES
      ELSEIF (CLINE(1:13).EQ.'MESSAGE ENTRY') THEN
        NMESLNS = NMESLNS+1
        IF (NC.GE.15) MESTXT(NMESLNS) = ALINE(15:NC)
      ELSEIF (CLINE(1:23).EQ.'MESSAGE POSITION, ANGLE') THEN
        IF (NMESLNS.EQ.0) GO TO 1
        READ (ALINE(25:NC),*,ERR=1) XMESF(NMESLNS),YMESF(NMESLNS),AN-
GMES(NMESLNS)
* plotlegend *
      ELSEIF (CLINE(1:15).EQ.'LEGEND POSITION') THEN
        READ (ALINE(17:NC),*,ERR=1) XLEGF,YLEGF
      ELSEIF (CLINE(1:11).EQ.'LEGEND SIZE') THEN
        READ (ALINE(13:NC),*,ERR=1) SIZEG
      ELSEIF (CLINE(1:12).EQ.'LEGEND TITLE') THEN
        IF (NC.GE.14) LEGTIT = ALINE(14:NC)
      ELSEIF (CLINE(1:12).EQ.'LEGEND ENTRY') THEN
        IF (NC.GE.14) LEGTXT(NCURV) = ALINE(14:NC)
* plotlines *
      ELSEIF (CLINE(1:14).EQ.'LINE COLOR RED') THEN
        LCOLOR(NCURV) = 1
      ELSEIF (CLINE(1:16).EQ.'LINE COLOR GREEN') THEN
        LCOLOR(NCURV) = 2
      ELSEIF (CLINE(1:15).EQ.'LINE COLOR BLUE') THEN
        LCOLOR(NCURV) = 3
      ELSEIF (CLINE(1:18).EQ.'LINE COLOR MAGENTA') THEN
        LCOLOR(NCURV) = 4
      ELSEIF (CLINE(1:17).EQ.'LINE COLOR YELLOW') THEN
        LCOLOR(NCURV) = 5
      ELSEIF (CLINE(1:15).EQ.'LINE COLOR CYAN') THEN
        LCOLOR(NCURV) = 6
      ELSEIF (CLINE(1:16).EQ.'LINE COLOR WHITE') THEN
        LCOLOR(NCURV) = 7
      ELSEIF (CLINE(1:14).EQ.'LINE TYPE BOLD') THEN
        LBOLD(NCURV) = 1
      ELSEIF (CLINE(1:22).EQ.'LINE TYPE SYMBOLS ONLY') THEN
        LTYPE(NCURV) = -2
      ELSEIF (CLINE(1:27).EQ.'LINE TYPE CONNECTED SYMBOLS') THEN
        LTYPE(NCURV) = -1
      ELSEIF (CLINE(1:16).EQ.'LINE TYPE DASHED') THEN
        LTYPE(NCURV) = 1
      ELSEIF (CLINE(1:16).EQ.'LINE TYPE DOTTED') THEN
        LTYPE(NCURV) = 2
      ELSEIF (CLINE(1:22).EQ.'LINE TYPE CHAIN-DASHED') THEN
        LTYPE(NCURV) = 3
      ELSEIF (CLINE(1:22).EQ.'LINE TYPE CHAIN-DOTTED') THEN
        LTYPE(NCURV) = 4
      ELSEIF (CLINE(1:13).EQ.'SYMBOL NUMBER') THEN
        READ (ALINE(15:NC),*,ERR=1) LSYMBOL(NCURV)
      ELSEIF (CLINE(1:11).EQ.'SYMBOL SKIP') THEN
        READ (ALINE(13:NC),*,ERR=1) LSKIP(NCURV)
      ELSEIF (CLINE(1:11).EQ.'SYMBOL SIZE') THEN
        READ (ALINE(13:NC),*,ERR=1) SIZSYM
* plotlabel *
      ELSEIF (CLINE(1:14).EQ.'TITLE POSITION') THEN

```

```

      READ (ALINE(16:NC),*,ERR=1) XTITF,YTITF
    ELSEIF (CLINE(1:10).EQ.'TITLE SIZE') THEN
      READ (ALINE(12:NC),*,ERR=1) SIZTIT
    ELSEIF (CLINE(1:10).EQ.'PLOT TITLE') THEN
      IF (NC.GE.12) TITLE = ALINE(12:NC)
    ELSEIF (CLINE(1:15).EQ.'AXIS LABEL SIZE') THEN
      READ (ALINE(17:NC),*,ERR=1) SIZLAB
    ELSEIF (CLINE(1:12).EQ.'X AXIS LABEL') THEN
      IF (NC.GE.14) XLBL = ALINE(14:NC)
    ELSEIF (CLINE(1:12).EQ.'Y AXIS LABEL') THEN
      IF (NC.GE.14) YLBL = ALINE(14:NC)
    ELSEIF (CLINE(1:11).EQ.'FONT TRIPLX') THEN
      IFONT = 1
    ELSEIF (CLINE(1:18).EQ.'FONT SWISSL SHADED') THEN
      IFONT = 2
      ISHAD = 1
    ELSEIF (CLINE(1:17).EQ.'FONT SERIF SHADED') THEN
      IFONT = 3
      ISHAD = 1
    ELSEIF (CLINE(1:11).EQ.'FONT SWISSL') THEN
      IFONT = 2
    ELSEIF (CLINE(1:10).EQ.'FONT SERIF') THEN
      IFONT = 3
* plotaxes *
    ELSEIF (CLINE(1:30).EQ.'X AXIS MIN MAX STEP MULTIPLIER') THEN
      IXS = 1
      READ (ALINE(32:NC),*,ERR=1) XMN,XXM,XSTEP,XXMULT
    ELSEIF (CLINE(1:10).EQ.'X AXIS LOG') THEN
      LOGX = 1
    ELSEIF (CLINE(1:30).EQ.'Y AXIS MIN MAX STEP MULTIPLIER') THEN
      IYS = 1
      READ (ALINE(32:NC),*,ERR=1) YMN,YMY,YSTEP,YMULT
    ELSEIF (CLINE(1:10).EQ.'Y AXIS LOG') THEN
      LOGY = 1
* plotsize *
    ELSEIF (CLINE(1:21).EQ.'X-Y ORIGIN X-Y LENGTH') THEN
      READ (ALINE(23:NC),*,ERR=1) XORIGIN,YORIGIN,XLENIN,YLENIN
* data point *
    ELSE
5      IP=IP+1
      IF (IP.GT.MAXPTS) THEN
        WRITE (*,
: ' ('' Maximum number of points ('',I4,'') exceeded.'')' ) MAXPTS
        ERRORS=.TRUE.
        RETURN
      ENDIF
      READ (ALINE,*,ERR=1) X(IP),Y(IP)
      NP (NCURV)=NP (NCURV)+1
      XYDATA=.TRUE.
    ENDIF
    GO TO 2
3  CONTINUE
    RETURN
1  WRITE (*,' ('' Error in line '',I3,'': '',80A)')
: ILINE,ALINE(1:NC)
    ERRORS = .TRUE.
    RETURN
  END

```

Appendix E

Student's "t" Program and Scientific Uncertainty Spreadsheet Format Example

```
*****
*      Program to compute the Student's t95 value.      *
*      DBETAI is a SLATEC double precision subroutine that uses the *
*      Incomplete Beta Function, I, which is used to compute t95.  *
*****
PROGRAM STUDENTT
IMPLICIT REAL*8 (A-H,O-Z)
COMMON CONFLEV, DNU
EXTERNAL PROBmCON
CONFLEV = .95D0
OPEN (UNIT=7,FILE='STUDENTT.DAT')
DO NU = 1,100
  DNU = 1.0d0*NU
  STUDTT95 = RTBIS(PROBmCON, 1.9D0, 13.D0, 1.D-12)
  write(6,'(i6,3x,f12.9)') NU, STUDTT95
  write(7,'(i6,3x,f12.9)') NU, STUDTT95
ENDDO
DO NU = 200, 1000, 100
  DNU = 1.0d0*NU
  STUDTT95 = RTBIS(PROBmCON, 1.0D0, 13.D0, 1.D-12)
  write(6,'(i6,3x,f12.9)') NU, STUDTT95
  write(7,'(i6,3x,f12.9)') NU, STUDTT95
ENDDO
DO NU = 2000, 10000, 1000
  DNU = 1.0d0*NU
  STUDTT95 = RTBIS(PROBmCON, 1.0D0, 13.D0, 1.D-12)
  write(6,'(i6,3x,f12.9)') NU, STUDTT95
  write(7,'(i6,3x,f12.9)') NU, STUDTT95
ENDDO
DO NU = 20000, 100000, 10000
  DNU = 1.0d0*NU
  STUDTT95 = RTBIS(PROBmCON, 1.0D0, 13.D0, 1.D-12)
  write(6,'(i6,3x,f12.9)') NU, STUDTT95
  write(7,'(i6,3x,f12.9)') NU, STUDTT95
ENDDO
DO NU = 200000, 1000000, 100000
  DNU = 1.0d0*NU
  STUDTT95 = RTBIS(PROBmCON, 1.0D0, 13.D0, 1.D-12)
  write(6,'(i6,3x,f12.9)') NU, STUDTT95
  write(7,'(i6,3x,f12.9)') NU, STUDTT95
ENDDO
END
```

```

FUNCTION PROBmCON (T)
IMPLICIT REAL*8 (A-H,O-Z)
COMMON CONFLEV, DNU
PARAMETER (QIN = 0.5D0)
X = DNU/(DNU + T*T)
PIN = DNU/2.0D0
PROB = 1.0D0 - DBETAI(X, PIN, QIN)
PROBmCON = PROB - CONFLEV
RETURN
END

```

```

FUNCTION RTBIS(FUNC,X1,X2,XACC)
IMPLICIT REAL*8 (A-H,O-Z)
PARAMETER (JMAX=50)
FMID=FUNC(X2)
F=FUNC(X1)
IF (F*FMID.GE.0.D0) PAUSE 'Root must be bracketed for bisection.'
IF (F.LT.0.D0) THEN
    RTBIS=X1
    DX=X2-X1
ELSE
    RTBIS=X2
    DX=X1-X2
ENDIF
DO 11 J=1,JMAX
    DX=DX*.5D0
    XMID=RTBIS+DX
    FMID=FUNC(XMID)
    IF (FMID.LE.0.D0) RTBIS=XMID
    IF (ABS(DX).LT.XACC .OR. FMID.EQ.0.D0) RETURN
11 CONTINUE
PAUSE 'too many bisections'
END

```

Performance Parameter	n	Precision	Bias	Uncertainty		Range	Notes
		Index	Limit	Uadd	Urss		
Test Result	7.83	4.47E-01	6.31E-01	9.79E-01	7.21E-01		

[illegible]

ELEMENTAL ERROR SOURCES (MEASUREMENT 1)

Elemental Error Source	Precision Index	Bias Limit	Substantiation Source and Notes
Calibration			
National -> Inter-Laboratory Standard	1.00E-02	2.00E-02	
Inter-Laboratory -> Transfer Standard	1.00E-02	2.00E-02	
Transfer -> Working Standard	1.00E-02	2.00E-02	
Working Standard -> Measurement Instrument	1.00E-02	2.00E-02	
Data Acquisition			
Excitation Voltage	1.00E-02	2.00E-02	
Electrical Simulation	1.00E-02	2.00E-02	
Signal Conditioning	1.00E-02	2.00E-02	
Recording Device	1.00E-02	2.00E-02	
Pressure Transducer	1.00E-02	2.00E-02	
Probe Errors	1.00E-02	2.00E-02	
Environmental Effects	1.00E-02	2.00E-02	
Data Reduction			
Curve Fit	1.00E-02	2.00E-02	
Computer Resolution	1.00E-02	2.00E-02	
Round Off	1.00E-02	2.00E-02	
Truncation	1.00E-02	2.00E-02	

ELEMENTAL ERROR SOURCES (MEASUREMENT 2)

Elemental Error Source	Precision Index	Bias Limit	Substantiation Source and Notes
------------------------	-----------------	------------	---------------------------------

Calibration

National -> Inter-Laboratory Standard	1.00E-02	2.00E-02	
Inter-Laboratory -> Transfer Standard	1.00E-02	2.00E-02	
Transfer -> Working Standard	1.00E-02	2.00E-02	
Working Standard -> Measurement Instrument	1.00E-02	2.00E-02	

Data Acquisition

Excitation Voltage	1.00E-02	2.00E-02	
Electrical Simulation	1.00E-02	2.00E-02	
Signal Conditioning	1.00E-02	2.00E-02	
Recording Device	1.00E-02	2.00E-02	
Pressure Transducer	1.00E-02	2.00E-02	
Probe Errors	1.00E-02	2.00E-02	
Environmental Effects	1.00E-02	2.00E-02	

Data Reduction

Curve Fit	1.00E-02	2.00E-02	
Computer Resolution	1.00E-02	2.00E-02	
Round Off	1.00E-02	2.00E-02	
Truncation	1.00E-02	2.00E-02	

ELEMENTAL ERROR SOURCES (MEASUREMENT 3)

Elemental Error Source	Precision Index	Bias Limit	Substantiation Source and Notes
------------------------	-----------------	------------	---------------------------------

Calibration

National -> Inter-Laboratory Standard	1.00E-02	2.00E-02	
Inter-Laboratory -> Transfer Standard	1.00E-02	2.00E-02	
Transfer -> Working Standard	1.00E-02	2.00E-02	
Working Standard -> Measurement Instrument	1.00E-02	2.00E-02	

Data Acquisition

Excitation Voltage	1.00E-02	2.00E-02	
Electrical Simulation	1.00E-02	2.00E-02	
Signal Conditioning	1.00E-02	2.00E-02	
Recording Device	1.00E-02	2.00E-02	
Pressure Transducer	1.00E-02	2.00E-02	
Probe Errors	1.00E-02	2.00E-02	
Environmental Effects	1.00E-02	2.00E-02	

Data Reduction

Curve Fit	1.00E-02	2.00E-02	
Computer Resolution	1.00E-02	2.00E-02	
Round Off	1.00E-02	2.00E-02	
Truncation	1.00E-02	2.00E-02	

Appendix F

Space Systems Network User's Guide

This appendix contains the Space Systems Network User's Guide prepared to properly document the changes made to the Space Systems Ethernet Network and interconnected VAX systems to support the FPA testing as part of the CN45VW effort. The user's guide format and pagination was selected to specifically enhance visual tracking and to facilitate future updates as additions and changes are made to individual sections. The User's Guide should be updated periodically as the computational hardware and software resources evolve.

Space Systems Network User's Guide

Sidney Steely / January 1991

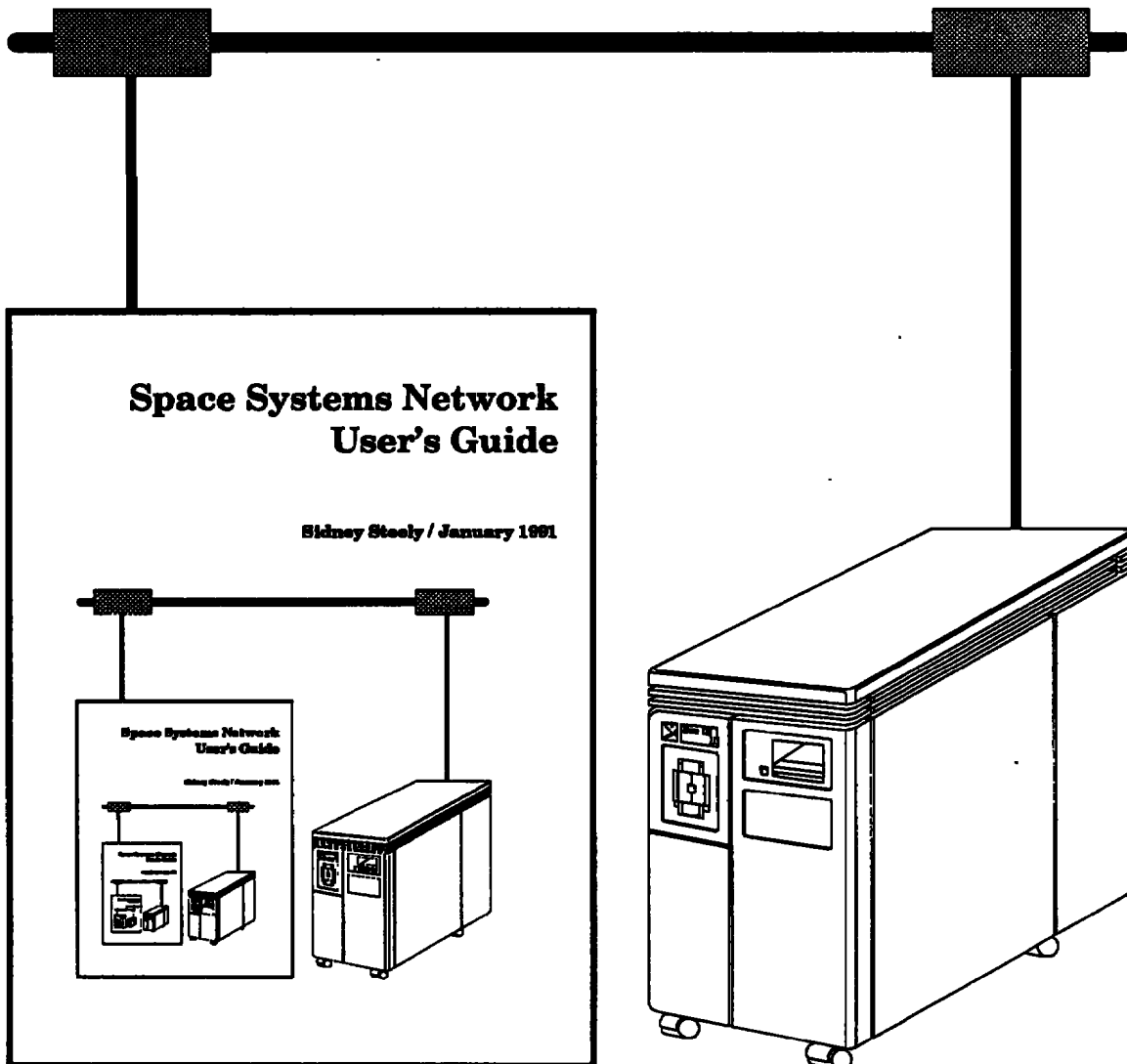


Table of Contents

1 INTRODUCTION

2 HARDWARE DESCRIPTION

2.1 NETWORK CONFIGURATION 2-2

2.2 PRIMARY COMPONENTS 2-3

2.2.1 MICROVAX II 2-3

2.2.2 MicroVAX III (FPAMIC) 2-4

2.2.3 DELNI 2-5

2.2.4 DEMPER 2-5

2.2.5 DECSERVERS 2-6

2.2.6 SYSTEM PRINTERS 2-6

2.2.6.1 LN03-PLUS (Regular Laser Printer) 2-7

2.2.6.2 LN03R (Postscript Laser Printer) 2-7

2.2.6.3 Additional Printers 2-7

2.2.7 LVP16 SIX-PEN COLOR PLOTTER 2-8

2.2.8 HP7550 EIGHT-PEN COLOR PLOTTER 2-8

2.2.9 SYSTEM TERMINALS 2-8

2.2.9.1 SYSTEM CONSOLE TERMINALS 2-8

2.2.9.2 TERMINAL-SERVER TERMINALS 2-8

2.3 OTHER ETHERNETED SYSTEMS 2-9

2.3.1 DEL 310 (node FPAOI) 2-9

2.3.2 MicroVAX III (node SGTC) 2-9

2.3.3 VAXstation III/GPXs (nodes ELAGPX and SIDGPX) 2-9

3 MANAGEMENT/OPERATION

3.1 USER ACCOUNTS AND RESOURCES 3-1

3.2 USER ACCOUNTS AND SECURITY 3-1

3.3 DISK STORAGE 3-2

3.3.1 SYSTEM DISKS 3-3

3.3.2 USER AND THIRD-PARTY SOFTWARE DISKS 3-3

3.4 DISK BACKUPS AND RESTORATION 3-4

4 SYSTEM USAGE

4.1 GENERAL ACCESS	4-1
4.2 USER ACCOUNTS	4-1
4.2.1 NEW USER ACCOUNTS	4-1
4.2.2 DISABLED ACCOUNTS	4-2
4.3 SYSTEM ACCESS	4-2
4.3.1 SYSTEM CONSOLE MONITORS	4-2
4.3.2 TERMINAL SERVERS	4-2
4.3.3 REMOTE NETWORK NODES	4-5
4.4 PRINTER ACCESS	4-5
4.4.1 REGULAR PRINTER (LN03-PLUS)	4-5
4.4.2 POSTSCRIPT PRINTER (LN03R)	4-7
4.4.3 OTHER NETWORKED PRINTERS	4-7

5 SOFTWARE DESCRIPTIONS

5.1 DEC SYSTEM SOFTWARE	5-1
5.1.1 VMS OPERATING SYSTEM	5-1
5.1.2 DECNET-VAX	5-2
5.1.3 DECSERVER 100/200	5-2
5.1.4 LATPLUS	5-3
5.2 DEC GENERAL-USE LAYERED PRODUCTS	5-3
5.2.1 VAX FORTRAN	5-3
5.2.2 VAX C	5-4
5.3 THIRD PARTY SOFTWARE	5-4
5.3.1 CALC	5-4
5.3.2 CODEV	5-5
5.3.3 DISSPLA	5-6
5.3.4 GUERAP	5-7
5.3.5 ISIS	5-7
5.3.6 KERMIT	5-8
5.3.7 OPTICAL SIGNATURE CODE	5-9
5.3.8 POLYPAGOS	5-9
5.3.9 SLATEC	5-10
5.3.10 TIGER	5-11
5.3.11 VISAGE	5-11
5.3.12 WORDMARC COMPOSER +	5-12

1 INTRODUCTION

The Space Systems Network is primarily an Ethernet computer network of computer systems and currently consists of several MicroVAX and VAXstation computer systems interconnected by thick- and thin-wire Ethernet hardware and software. The network hardware and interconnected computer systems are located at Arnold Air Force Base in the Engineering Laboratory Addition (ELA). The network system serves a variety of scientific functions in support of Space Systems analysis, technology, and testing activities. This includes many forms of data acquisition/reduction, scientific analysis and technical research which can be rendered solvable using VAX/VMS class of computers.

This User's Guide is an attempt to provide most of the local hardware and software information necessary to facilitate a better understanding of the analytical/computational tools available to support some of the Space Systems Branch scientific efforts. Special emphasis has been devoted to explaining the networking configuration, specific hardware and software available, and local additions and changes to the operating system. Hopefully, this information will ease and enhance the use of the networked computer systems and render scientific users a more efficient working environment. It is hoped that this information proves to be useful to the novice as well as the experienced users.

Users should be at least aware of the comprehensive set of Digital Equipment Corporation (DEC) Reference Manuals, User Guides, and the Software Training tutorials available in the Computational Work Center. This "Space Systems Network User's Guide" does not duplicate or replace the wealth of Digital Documentation provided for general use. This user's guide serves only as a supplement to help orient users and to explain the "network system" as it is currently configured. Users should especially become familiar with the online help documentation provided with each computer node by simply typing HELP when logged into and using one of the computer system nodes; the HELP information will provide answers to most of your operating-system related questions. As additions or changes are made to the network

system an attempt will be made to inform users and to update this locally-specific user's guide.

As a result of very recent hardware and software additions/changes to support Focal Plane Array testing, this User's Guide has been written to facilitate updates to individual sections. The dynamic evolution of the network system's hardware and software configuration makes it difficult to maintain current user information. Section updates will be provided as deemed necessary, required, and/or useful (time permitting). The Space Systems network configuration was reconfigured to support new focal plane array testing needs. This user's guide was prepared as a result of the new configuration and numerous software enhancements provided by and for the CN45VW Space Systems Analysis Support project during FY89-90.

The following major sections describe some of the specific aspects of our networked system. Section 2 introduces the various hardware components that constitute "the network system" (including networking components). Section 3 discusses general management of the system and normal modes of operation. Section 4 explains how to access and use the network system. Section 5 provides a summary explanation of the current software on the system for general-use purposes.

Several appendices have been included at the end. Each appendix describes detailed information about specific software packages and their attributes frequently requested by users and deemed generally important, but not otherwise readily available to users and too detailed for inclusion in the more general software summaries provided in Section 5.

2 HARDWARE DESCRIPTION

The current Space Systems Ethernet components have been reconfigured to provide maximum functionality yet a flexible path for future growth. The network configuration was modified to provide support for the Focal Plane Array Testing effort. To maximize available resources and provide the necessary path for growth, the system was originally configured as a local area network for three users sharing only a single computer system. The number of users has grown to over forty while the hardware has evolved to continue general scientific support; some dedicated computer systems and several terminal servers have also been added. The "network system" and its major components are illustrated in Fig. 2-1, the major elements will be described in the following sections.

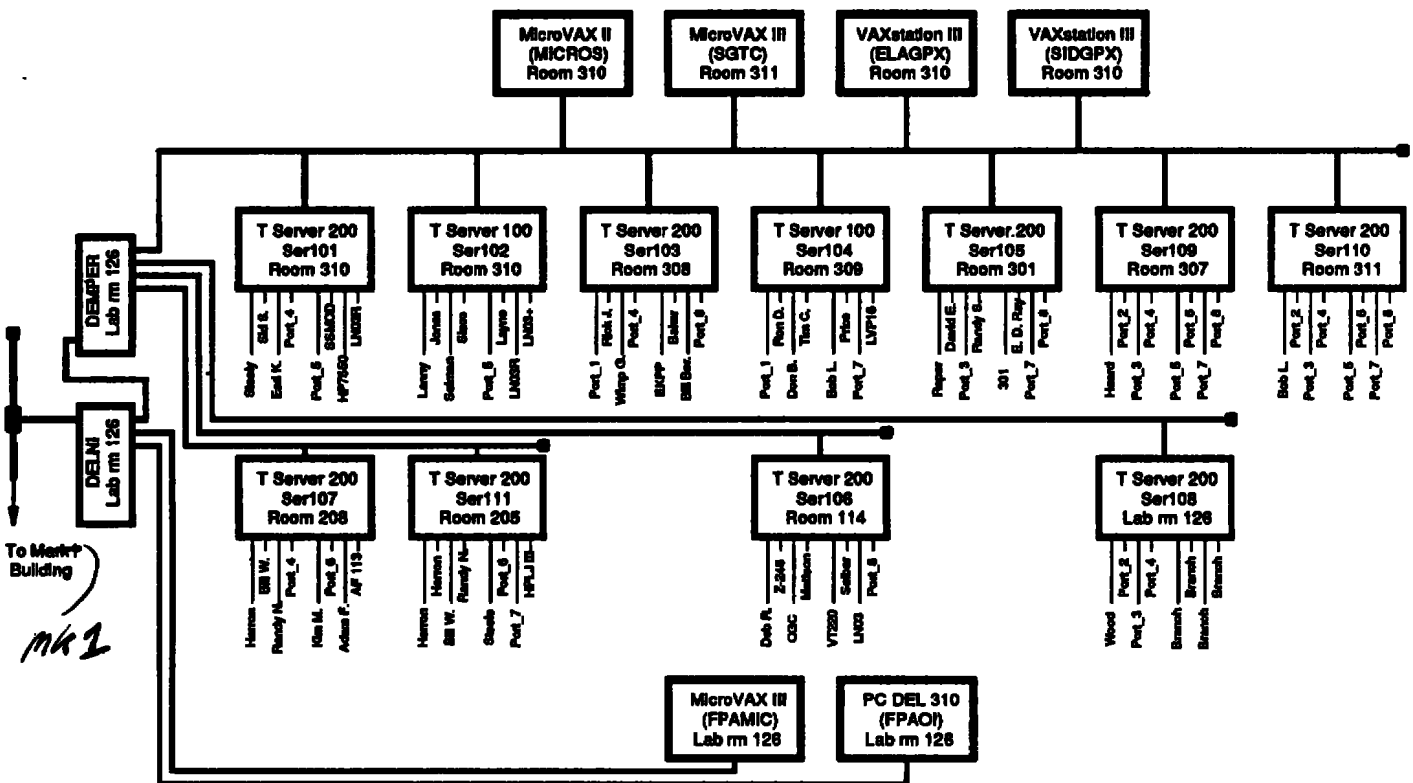


Figure 2-1. Network configuration and major components.

2.1 NETWORK CONFIGURATION

Ethernet is Digital's Local Area Network (LAN). It provides fast, efficient, and transparent communication and data exchange; a flexible cpu-off-loading-system for user interconnects; peripheral and resource sharing for cost savings; and an excellent path for future expansions as needed.

From Fig. 2-1 we see that the system is "glued" together with Digital Ethernet components; although a matter of opinion, some researchers consider the network as "the system." The Ethernet communication links come in two primary flavors, backbone thick-wire coaxial cable and the thin-wire variant. The original 1985 Space Systems Ethernet local area network installation consisted only of 10 million bps thick-wire coaxial cable, H4000's and a Local Area Network Interconnect (DELNI).

Although the stand-alone and hierarchical stand-alone DELNI LAN eliminates the need for a coaxial cable and an H4000 transceiver, the thick-wire coaxial line has been installed between the ELA-1077 and ^{mk 1} MARK1 buildings to provide access to the existing VAX 8650 facility computer system. It is seen from Fig. 2-1 that a thick-wire data interconnection goes from the DELNI to the ^{mk 1} MARK1 building; transparent to most users, this connection is via the DELNI, two H4000 transceivers, and an interlinking thick-wire coaxial cable.

Since the original installation, an additional MicroVAX III (FPAMIC) and two VAXstation III/GPX color graphics workstation (node names ELAGPX and SIDGPX) have been installed in the ELA building. These nodes are interconnected via the DELNI and Digital Ethernet Multiport Repeater (DEMPER). The MicroVAX III (FPAMIC) is dedicated to supporting focal-plane-array testing.

No attempt will be made to explain, in detail, the individual network components since their operation is generally transparent to most users. Users interested in the networking components, the operational constraints, and modus operandi should consult the "Digital Networks and Communications Buyer's Guide and related DECnet and LAVC documentation." (The Terminal servers will, however, require additional explanation in Section 4.)

2.2 PRIMARY COMPONENTS

Four computer network nodes (ELAGPX, FPAMIC, MICROS, and SIDGPX) have been clustered in an Ethernet-based Local Area VAX Cluster (LAVC) to optimize their use and disk storage. This configuration was selected to support the new FPA testing and technology needs. The reconfiguration helps to reduce the proliferation and redundancy of unnecessary user files and helps users by having to only maintain one set of files in their working space. A single DEL 386 computer system (node name FPAOI) has been added to the Ethernet to support the FPA testing interface. One additional temporary network node has also been added to support the SGTC Direct Write Scene Generation (DWSG) effort. The general-user support nodes include MICROS and FPAMIC.

2.2.1 MICROVAX II

The general user support node MICROS is a MicroVAX II computer system and consists of a 32-bit scientific processor/coprocessor and several components which have been configured to provide optimum system performance and commonality/compatibility with current and future systems. The MicroVAX II computer currently contains several standard Digital elements:

- A BA123 System Housing and Power Supply
- KA-630 CPU and Coprocessor Board (includes 1 MB of memory)
- 2-MB Memory Board
- 4-MB Memory Board
- DEQNA Ethernet Controller Board
- DHV11 Eight-Line RS232 Interface Board (TXA_)
- TK50 Tape Controller Board (MUA0:)
- RQDX3 Disk Controller Board (DUA_:) (Primary)
- RQDX2 Disk Controller Board (DUB_:) (Secondary)
- RD53 Digital 71 MB Hard Disk (4) (DUA0: - DUA3:)
- RD53 Digital 71 MB Hard Disk (DUB0:)
- RX50 Floppy Disk drive (DUB1: and DUB2:)
- TK50 95 MB Tape Drive (MUA0:)

The RX50 and TK50 drives have been removed from the MicroVAX II housing and installed in a tightly-coupled external housing to

allow two additional disk drives to be installed in the BA123 system housing. Use of the RX50 and TK50 is the same as before and as described in the Digital MicroVMS User's Guide.

The MICROS node is connected to the Ethernet LAN using the internal DEQNA controller board, a transceiver communication cable, and thin-wire Ethernet cable. This computer has been assigned the network node name MICROS and supports user access by a direct connection to the DHV11 interface, remote nodes, or by the Ethernet terminal servers as illustrated in Fig. 2-1. Primary access is, however, via the individual terminal servers, which will be described later.

2.2.2 MicroVAX III (FPAMIC)

The computer node FPAMIC is a MicroVAX III computer system. It primarily supports users for the FPA testing effort and consists of a 32-bit scientific processor/coprocessor and several components which have been configured to provide optimum system performance and commonality/compatibility with current and future systems. The MicroVAX III computer currently contains several standard Digital elements:

- A H9642-J System Cabinet and Power Supply
- KA650 CPU/Coprocessor Board
- 8-MB Memory Board (2) (16 Megabytes total)
- Two DEQNA Ethernet Controller Boards
- DHV11 Eight-Line RS232 Interface Board (TXA_)
- DZQ11 Four-Line RS232 Interface Board (TTA_)
- Parallel Printer Port (LPA0:)
- TK50 Tape Controller Board (MUA0:)
- KDA50 Disk Controller Board (DUB_) (Primary Controller)
- RA81 Hard Disk (DUB0:, System Disk)
- RQDX3 Disk Controller Board (DUA_) (Secondary Controller)
- Two RD53 Digital 71 MB Hard Disk (3) (DUA0: and DUA1:, User Disk)
- TK50 95 MB Tape Drive (MUA0:)

The FPAMIC node is connected to the Ethernet LAN using the internal DEQNA controller board, a transceiver communication cable, and the DELNI as illustrated in Fig 2-1. This computer has been assigned the network node name FPAMIC and supports user access primarily by Ethernet terminal servers.

2.2.3 DELNI

The Digital Ethernet Local Network Interconnect (DELNI) is a low-cost table-top concentrator device that allows up to eight Ethernet compatible devices to be grouped together in a relatively small local area network. Whether the system devices are connected together using a DELNI or thick-wire coaxial cable with H4000 transceivers, the throughput performance is the same, i.e. 10 Mbps. The DELNI does provide the potential isolation (switchable) to run independent of any thick-wire or thin-wire interconnections. The DELNI can be operated in three modes: stand-alone, hierarchical stand-alone, or connected. A switch on the DELNI allows stand-alone or connected (global) modes of operation. The stand-alone operation allows for temporary isolation of network traffic in the DELNI subnet for diagnostics, performance testing, or potential security modes of operation. From Fig. 2-1 we see that the MicroVAX III can operate in an isolated mode when the need arises.

2.2.4 DEMPER

Thin-wire Ethernet offers an alternative cabling system that provides full Ethernet capability for workstations, personal computers, and small localized working groups. In particular, the thin-wire is RG58 C/U-type, flexible, and easily installed coaxial cable. System devices are easily connected to the thin-wire with standard BNC-type T-connector junctions.

The Digital Ethernet Multiport Repeater (DEMPER) is similar to the DELNI in that it allows multiple (up to eight) segments of thin-wire-connected systems to be interconnected. Each of the eight thin-wire segments can be up to 185 meters long and can support up to 29 devices (30 including the DEMPER). The DEMPER also contains one port for a transceiver to be connect to standard Ethernet thick-wire coax via an H4000 transceiver or via the DELNI as is currently configured and illustrated in Fig. 2-1. Not shown in Fig. 2-1 are thin-wire Ethernet Station Adapters (DESTA) and t-connectors that are required for each of the devices connected to the thin-wire Ethernet segments.

terminal server, DECserver node SERV02 located in the ELA building room 310. The following two sections describe the current printer hardware connections.

2.2.6.1 LN03-PLUS (Regular Laser Printer)

An LN03-PLUS 8 page per minute Digital Equipment Corporation laser printer is connected to PORT_8 of the DECserver 100 terminal server (node SER102). Each node's logical port LTA1: is initialized when the node is powered up and booted. The logical port LTA1: is initialized in the SYS\$MANAGER:LTLOAD.COM command file on bootup; LTA1: is logically mapped to the terminal server's SER102 port PORT_8. The SYS\$PRINT and SYS\$LN03PLUS1 print queues are initialized and started on bootup in the SYS\$MANAGER:REMOTE_PRINT.COM command file. The default print output of the nodes are then directed to SYS\$PRINT, indirectly through SYS\$LN03PLUS1, and ultimately ends up at the LN03-PLUS laser printer. This interlink is generally transparent to most users and provides the flexibility for all LAVC nodes to have a similar logical port and print queue defined to share this printer resource; the DECserver also queues the different nodes' print requests to the LN03-PLUS printer. The LN03-PLUS laser printer servers as a text and graphics printer. The LN03-PLUS printer has a Tektronix 4014 emulation mode to support graphics output (DISSPLA, etc.).

2.2.6.2 LN03R (Postscript Laser Printer)

In a similar manner, a Digital Equipment Corporation LN03R postscript laser printer is connected to PORT_7 of the DECserver 100 (node SER102), see Fig. 2-1. Each node's logical port LTA2: is initialized and started in a manner similar to LTA1:. A special print queue, SYS\$LN03R, has also been created to support postscript output. This printer is available for any node on the Ethernet and general use.

2.2.6.3 Additional Printers

Several additional laser printers are also located on the Ethernet terminal servers for direct support and special applications. Printer queues have been established to print to these devices as needed. If the regular default system printers are inadequate you should see the system manager for additional information concerning their use.

2.2.7 LVP16 SIX-PEN COLOR PLOTTER

A Digital LVP16 six-pen color plotter is connected to the logical port LTA1: which is mapped to the terminal server SER104 PORT_8 (see Fig. 2-1). The plotter is functionally equivalent to an HP7475 plotter. The plotter uses the HPGL protocol graphics language. The plotter is located in the ELA room 309.

2.2.8 HP7550 EIGHT-PEN COLOR PLOTTER

Similarly, an HP7550 eight-pen color plotter is connected to the logical port LTA3: which is mapped to the terminal server SER101 PORT_7 (see Fig. 2-1). The plotter also uses the HPGL protocol graphics language and is a sheet-feed color pen plotter. The plotter is located in the ELA room 310.

2.2.9 SYSTEM TERMINALS

2.2.9.1 SYSTEM CONSOLE TERMINALS

Each MicroVAX node's console terminal (connected to the console port) is primarily for booting the system, system messages, installing software and general system control and operation. Users can use this terminal but should be aware that system messages are directed to the console port and may be distracting or inconvenient to use. Many MicroVAX network and system problems require direct access to the MicroVAX console terminals; their general use is discouraged.

2.2.9.2 TERMINAL-SERVER TERMINALS

The DECserver terminal servers provide the primary method of accessing the network and LAVC computer systems. The terminal servers are connected to the Ethernet as illustrated in Fig. 2-1. The terminal servers provide a semi-direct connection to any computer on the LAN. Each server provides terminal connections which provide access to the desired resource. Most users should be aware of the terminal servers' operation and their salient features to improve their usefulness. The more important aspects of the terminal servers and their use are further discussed in Section 4.0 (System Usage).

2.3 OTHER ETHERNETED SYSTEMS

There are several other special purpose computer systems connected to the Ethernet. These Ethernet connections provide system communication, data exchange, and shared resources.

2.3.1 DEL 310 (node FPAOI)

The DEL 310 is an Intel 386/387-based computer system that was added to the Ethernet to support the Focal Plane Array testing effort. The system provides an interface to the FPA data acquisition systems and the MicroVAX III (FPAMIC) computer system. This is a single-user system dedicated to the FPA effort.

2.3.2 MicroVAX III (node SGTC)

The SGTC node is a MicroVAX III computer system that was added to the Ethernet for temporary support of the Direct Write Scene Generation effort. It is used to develop hardware and software interfaces (DECwindows, DataViews, etc.).

2.3.3 VAXstation III/GPXs (nodes ELAGPX and SIDGPX)

Two VAXstation III color graphics workstations have been added to the Ethernet network LAVC and are used for special graphics applications (CODEV optical analysis, DECwindows/X-windows interfaces, DISSPLA graphics output, VISAGE data analysis and display, etc.).

2 HARDWARE DESCRIPTION

The current Space Systems Ethernet components have been reconfigured to provide maximum functionality yet a flexible path for future growth. The network configuration was modified to provide support for the Focal Plane Array Testing effort. To maximize available resources and provide the necessary path for growth, the system was originally configured as a local area network for three users sharing only a single computer system. The number of users has grown to over forty while the hardware has evolved to continue general scientific support; some dedicated computer systems and several terminal servers have also been added. The "network system" and its major components are illustrated in Fig. 2-1, the major elements will be described in the following sections.

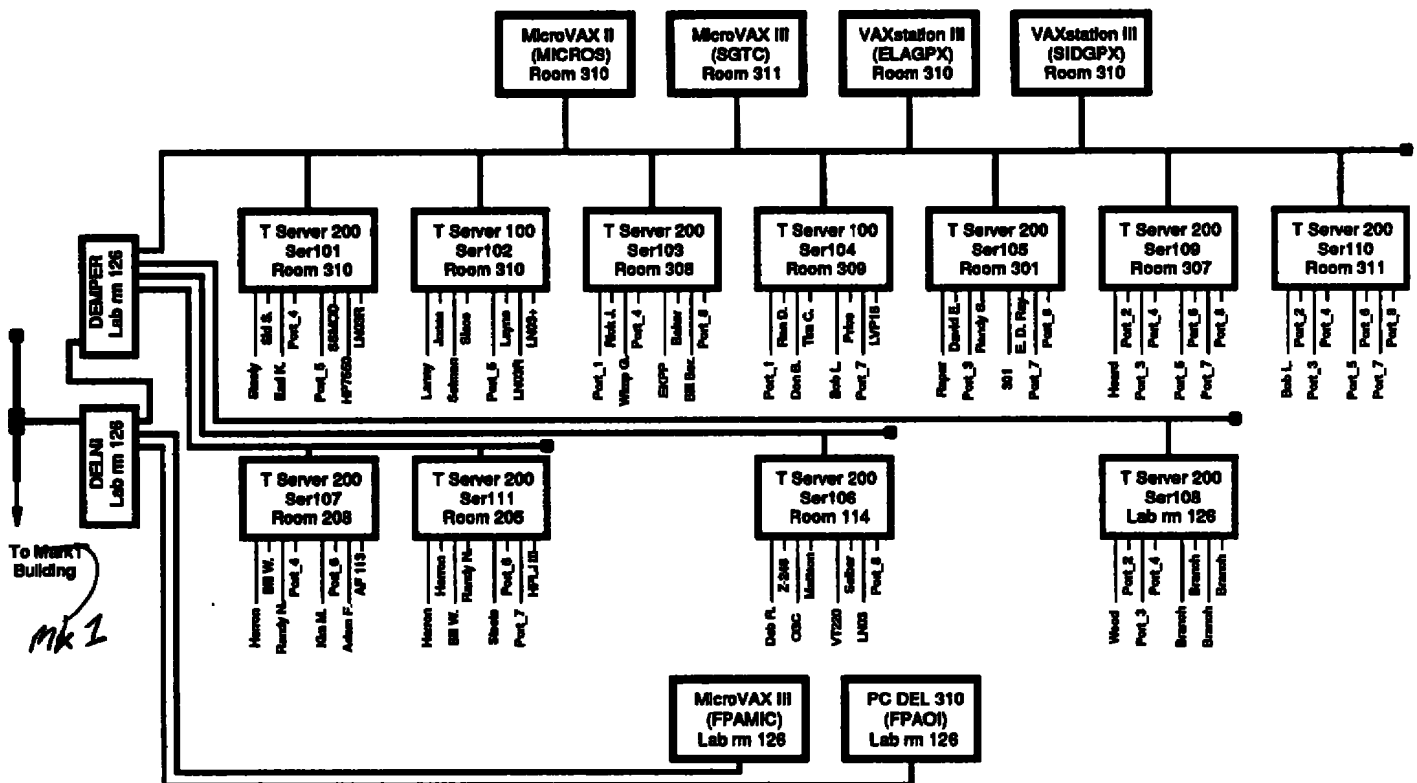


Figure 2-1. Network configuration and major components.

3 MANAGEMENT/OPERATION

The Space Systems nodes and network components are maintained and operated generally transparent to most users. It is, however extremely important to maintain and upgrade the hardware and software, providing an efficient and reliable computational resource for scientific users.

3.1 USER ACCOUNTS AND RESOURCES

All users generally gain access to the MicroVAX systems from their individual user accounts. New users are assigned an account on the node/system necessary to support their effort; each user must be assigned an account before he/she can access the desired computer node and computational resources. The user account provides users login access and file protection. Any new user needing additional information should contact the system manager.

3.2 USER ACCOUNTS AND SECURITY

User accounts are maintained by the system manager and stored in a protected area on each node in the node's SYS\$SYSTEM and SYS\$MANAGER workspace on the system. The user authorization file (UAF) is used by the system to validate login requests and to set up processes for users who successfully log in. Each UAF record consists of fields providing information for several areas: identification, login characteristics, login restrictions, priority, limits, and privileges.

The Space Systems Network nodes are currently operated in a non-secured (not System High) security mode. Future modes of operation will depend on the security requirements and needs.

Even though a System High Security mode is not in effect, sound judgment should still be used to provide protection to our system resources and to the individual users' work spaces to avoid catastrophic events, vandalism, and other types of system corruption.

The VMS operating system provides two related mechanisms to control the access that users have to system objects (files, disks, etc.):

- UIC-based protection -- Each user process in the system is assigned a user identification code (UIC) in the user authorization file . Each object on the system is also associated with a UIC (typically the UIC of the object creator). A protection mask associated with an object determines the type of access allowed to a user, based on the relationship between the user UIC and the object UIC.
- ACL-based protection -- An access control list (ACL) that specifies the type of access to be granted or denied to a particular user or group of users may be associated with a system object. The system objects for which ACL-based protection may be specified are: files, directories, devices, logical name tables, and global sections. Users are specified by identifiers in the rights database that are assigned with the Authorize Utility.

For a more complete description of the VMS protection schemes users should refer to the official Digital "VMS User's Manual" and documentation set provided in the computer work center.

UIC-based protection is the primary method used locally to protect user and system resources. Users in the Space Systems Branch have been normally assigned the same group UIC number 100 with a different user number component. This allows the Space Systems Branch users to interact as a working group and share files when needed. Users not in the Space System Branch have generally been assigned a different group number (not 100) for all of the users' privacy, protection, and security.

Each user account is assigned default parameters when created and the account is modified as required or needed to access abnormal resources. If you encounter problems with access or resources contact the system manager.

To generally help protect all user-created and system-created files and other UIC-protected objects, every user is requested to log off the system when finished with his/her work. Logging off is important and should be practiced to protect your own workspace and more importantly protect all the network nodes/systems.

3.3 DISK STORAGE

New users are generally assigned to one of the user disks and a disk quota set. Disk quotas are activated on the disks to provide

disk protection and to assure operational continuity. The LAVC networked systems have grown from a single node with only two disks to 4 nodes that share 13 disks currently.

3.3.1 SYSTEM DISKS

There are currently three disks used for system related activity and storage:

- **\$DISK1: (MICROS\$DUA0:)** -- This MICROS-resident disk is a 159 MB hard disk used to boot the system, pagefile/swapfile activity, and Digital Layered-Software storage.
- **\$DISK6: (ELAGPX\$DUA0:)** -- This ELAGPX-resident disk is a 159 MB hard disk used to boot the ELAGPX system, pagefile/swapfile activity, and Digital Layered-Software storage.
- **\$DISK11: (FPAMIC\$DUB0:)** -- This FPAMIC-resident disk is a 456 MB hard disk used to boot the FPAMIC system, pagefile/swapfile activity, and Digital Layered-Software storage.
- **\$DISK12: (SIDGPX\$DUA0:)** -- This SIDGPX-resident disk is a 159 MB hard disk used to boot the SIDGPX system, pagefile/swapfile activity, and Digital Layered-Software storage.

3.3.2 USER AND THIRD-PARTY SOFTWARE DISKS

There are currently two disks used for user related activity and storage:

- **\$DISK2: (MICROS\$DUA1:)** -- This disk is a 71 MB hard disk used for user files and user activity.
- **\$DISK3: (MICROS\$DUA2:)** -- This is a 71 MB hard disk used for third-party software storage.
- **\$DISK4: (MICROS\$DUA3:)** -- This is also a 71 MB hard disk used for user activity and storage.
- **\$DISK5: (MICROS\$DUB0:)** -- This is also a 71 MB hard disk used for user activity and storage.
- **\$DISK7: (ELAGPX\$DUA3:)** -- This is also a 71 MB hard disk used for user activity and storage.
- **\$DISK8: (ELAGPX\$DUA3:)** -- This is also a 71 MB hard disk used for user activity and storage.
- **\$DISK9: (FPAMIC\$DUA3:)** -- This is also a 71 MB hard disk used for user activity and storage.
- **\$DISK10: (FPAMIC\$DUA3:)** -- This is also a 71 MB hard disk used for user activity and storage.
- **\$DISK13: (SIDGPX\$DUA3:)** -- This is a 159 MB hard disk used for user activity and storage.

3.4 DISK BACKUPS AND RESTORATION

All system disk are backed up by the system managers to assure continuous operation with minimal impact from error in software installation or potential disk failure. The disks are periodically compressed to recover fragmented space, improve disk performance, and increase the MTBF. A new Diskkeeper software package has been installed on the Space Systems Network to compress the LAVC system disks daily for improved disk performance and to minimize disk failures.

All modified or new user disk files are backed up daily during the week to provide protection from catastrophic failures. End-of-month tape backups are retained but monthly tapes are rotated on a 12 month basis. No disks or files are archived for users; all users are expected to maintain their files and to retain a permanent copy on floppy or tape as desired or required. It should be reemphasized that the daily and monthly backups are rotated and do not serve as an archive. Our backups are generally to recover from a disk failure or accidental file deletion. Because the backup tapes are rotated, there is no guarantee that your file(s) will be saved on one of the monthly tapes unless that file existed at the time the tape backup was created and it was created within the past 12 months.

All disks are backed up at the end of each year, and the tapes are retained permanently. Unless your file existed at the time these backups were created you may not have a permanent backup copy.

All users should be aware of this backup policy and initiate their own archival/retention method; the Digital "VMS User's Manual" provides the necessary information describing this backup and archival process.

4 SYSTEM USAGE

The following sections provide specific information concerning the locally-peculiar aspects of the network interactions and special features added to assist new and experienced users in using the computational resources. Online help is an extremely important resource that new and experienced users should and will find useful to review new system features and to maintain current information; the online-help information is generally more current than most documentation users may have.

4.1 GENERAL ACCESS

New users gain access to the Space Systems Network nodes by acquiring an account on the desired system. See the node's system manager for information concerning the availability and use of a particular system.

A complete description of the VMS operating system is described in Digital's "Introduction to VAX/VMS" and more specific information in the Digital "VMS User's Manual". The documentation guides generally provide the novice and experienced users with adequate information to get around.

To avoid frustration and to maximize users' efficiency, it is highly recommended that each user take time to review the "VMS User's Manual" as soon as possible to acquire a basic working knowledge of the system. For most users this is only required once; the online-help facility will generally provide most of the specific information needed after a general understanding is acquired.

4.2 USER ACCOUNTS

4.2.1 NEW USER ACCOUNTS

Each new user of a Space Systems computer system is provided an account and disk space (quota) to store files and to conduct his

computational activity. All users access the computer nodes through their individual accounts, directly by a login session, or indirectly by a remote network connection.

New users are provided a template LOGIN.COM file in their workspace that can be modified to simplify many commands used frequently. New users can and should tailor their LOGIN.COM files at their discretion.

4.2.2 DISABLED ACCOUNTS

Users that resign, retire, or otherwise discontinue use of the system are removed from the Authorization List(s). The corresponding VAX system account(s) is deleted, and the user's directory and file structure are backed up for permanent retention (no guarantee) to make room for new-user disk space. If you intend to discontinue use of the network system, it is recommended that you backup your files to assure potential recovery should you decide to return or reactivate your account and restore your previous disk/file structure. See the System Manager if you have questions concerning this policy.

4.3 SYSTEM ACCESS

4.3.1 SYSTEM CONSOLE MONITORS

Terminals are connected to the MicroVAX console ports for system maintenance. This terminal provides a direct connection to the cpu interface port for booting the system and conducting system and operator activity. Because error, network, and operator messages are sent to this terminal, it is not generally available for use.

4.3.2 TERMINAL SERVERS

Most terminals (and some PCs) are connected to the terminal server ports as illustrated in Fig 2-1. Each of these terminals provides access to the computer nodes for general use. The operational aspects of the terminal servers are described in the "DECSERVER 100/200 Terminal Server User's Pocket Guide."

The terminal servers remain powered on and provide access to the network for general use. Most of the terminal server ports have been configured to connect to one of the network nodes, node MICROS is generally used by default. When the terminal server port has been set to a preferred node, entering the node name is not required. When prompted by the terminal server you will see the enter username prompt

Enter username> .

or the terminal server's local prompt

Local>

At the terminal server's username prompt enter your name as

Enter username> SMITH

and hit the return key to receive the system announcement and node's login prompt.

If the terminal server is displaying the local prompt then just type the connect command

Local> C (the quickest method)

or

Local> CONNECT

or

Local> CONNECT MICROS (node name is not required)

or

Local> C FPAMIC (C is short for CONNECT)

When finished with your login session just log out of the node, and the terminal server will return to the local mode prompt.

If your terminal, the server, and the desired node are operating properly and you have entered the correct information to initiate a login session on, for example, the MicroVAX II system, you will

see the system announcement and the username/password prompt as illustrated in Fig. 4-1 then the welcome information as illustrated in Fig. 4-2. The system announcement will generally contain information about down time, maintenance, etc.

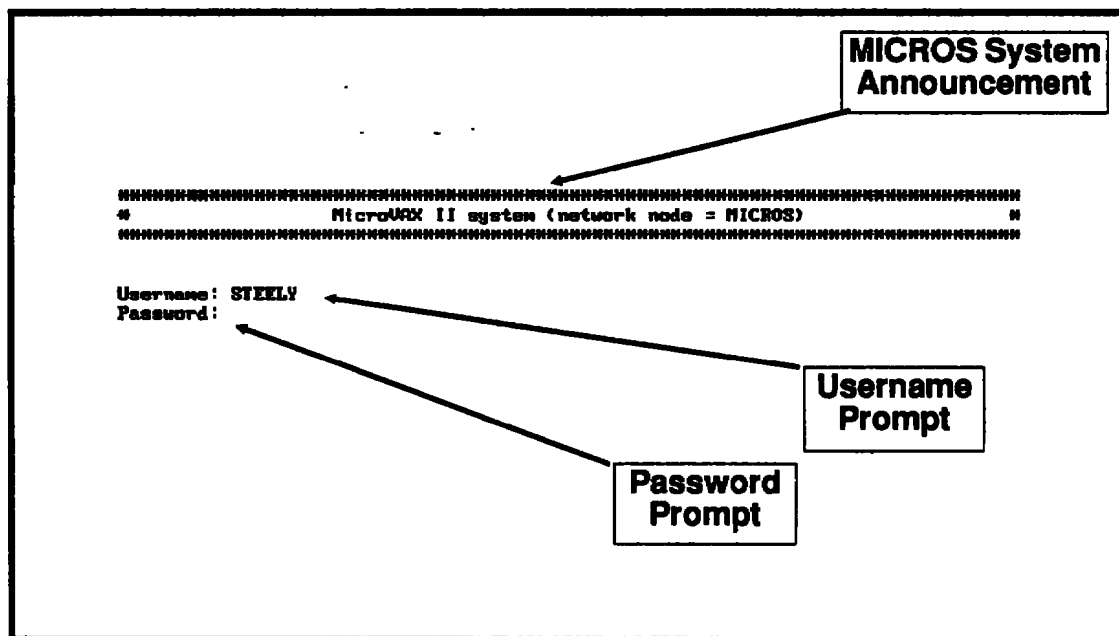


Figure 4-1. MicroVAX II announcement/username prompt.

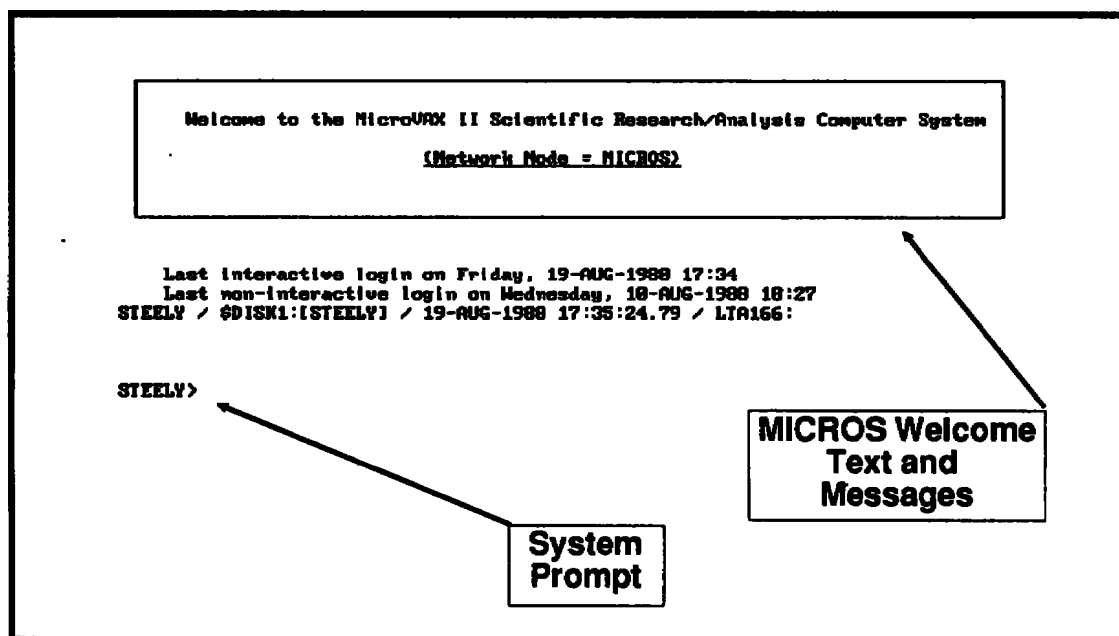


Figure 4-2. MicroVAX II welcome screen and messages.

After entering your username as illustrated in Fig. 4-1, the system will prompt you for your password. If you successfully enter a valid username/password combination the system welcome message will be displayed on your terminal as also shown in Fig. 4-2.

4.3.3 REMOTE NETWORK NODES

Any node that uses the DECNET software can communicate with other DECNET nodes. As we acquire other nodes (computers that can connect to the Ethernet), the networked systems can be logged into from the other nodes and visa versa. As an example, the SET HOST command could be executed from the VAXstation III/GPX (node ELAGPX) to connect to the MicroVAX II (node MICROS) by

```
$ SET HOST MICROS
```

The standard announcement and welcome messages will be displayed as described earlier. Just logout normally and your session will be returned to the originating node, ELAGPX in this case.

It is not always required to logon to another host node to use the available resources. Proxy access allow file transfers and other normal operations transparent to the user; you just append the node name onto your normal file specifications, for example

```
$ TYPE FPAMIC::$DISK10:[EARL]LOGIN.COM
```

would display user EARL's LOGIN.COM file on the screen if your node and the FPAMIC node was set for proxy access.

A full explanation of node-to-node communication and the SET HOST command is given in the Digital "VMS User's Manual".

4.4 PRINTER ACCESS ---

4.4.1 REGULAR PRINTER (LN03-PLUS)

New users are provided with some default symbols in their LOGIN.COM file to make printing a file simpler. Several print forms have been set up for the LN03-PLUS printer on the SYS\$LN03PLUS1 and SYS\$PRINT queues. The print forms pro-

vide a variety of print options that include formats used in the past prior to setting up the print forms. To use these forms you just include the /FORM switch on the PRINT command, e.g.

```
$ PRINT/FORM=3 filename.ext
```

will print your file using printer format number 3. If you type the show command

```
$ SHOW QUEUE/FORM
```

the current forms available will be displayed similar to Fig. 4-3.

\$ SH QUEUE/FORM		
Form name	Number	Description
DEFAULT	0	DEFAULT
POR_10CP1 (stock=DEFAULT)	1	POR_10CP1
POR_10CP1_BLRT (stock=DEFAULT)	2	POR_10CP1_BLRT
POR_10CP1_BT (stock=DEFAULT)	3	POR_10CP1_BT
POR_10CP1_L (stock=DEFAULT)	4	POR_10CP1_L
POR_10CP1_LR (stock=DEFAULT)	5	POR_10CP1_LR
POR_10CP1_LT (stock=DEFAULT)	6	POR_10CP1_LT
POR_12CP1 (stock=DEFAULT)	11	POR_12CP1
POR_12CP1_BLRT (stock=DEFAULT)	12	POR_12CP1_BLRT
POR_12CP1_BT (stock=DEFAULT)	13	POR_12CP1_BT
POR_12CP1_L (stock=DEFAULT)	14	POR_12CP1_L
POR_12CP1_LR (stock=DEFAULT)	15	POR_12CP1_LR
POR_12CP1_LT (stock=DEFAULT)	16	POR_12CP1_LT
POR_16CP1 (stock=DEFAULT)	21	POR_16CP1
POR_16CP1_BLRT (stock=DEFAULT)	22	POR_16CP1_BLRT
POR_16CP1_L (stock=DEFAULT)	23	POR_16CP1_L
POR_16CP1_LR (stock=DEFAULT)	24	POR_16CP1_LR
POSTSCRIPT (stock=DEFAULT)	51	POSTSCRIPT
\$		

Figure 4-3. Print forms for the LN03-PLUS printer.

The default print form is 0 for the SYS\$LN03PLUS1 printer queue and is not required when the PRINT command is entered. The default form is set for landscape format where the print is along the length of the paper. The default print characters are 16 cpi with 132 columns and 64 rows of characters. The other formats (except for 51, POSTSCRIPT) use a portrait printing orientation with a variety of character sizes (10, 12, and 16 cpi) and margins. The bottom, left, top, and right margin choices are indicated by the naming conventions BLRT, BT, L, LR, and LT. For example

```
$ PRINT/FORM=23 LOGIN.COM
```

will print the file LOGIN.COM using a portrait mode, 12 cpi character spacing, with only a left margin. This format is convenient for saving paper and increasing the print speed when a copy of your file is needed. Other utilities such as WORDMARC COMPOSER + can be used to select a variety of print formats also.

4.4.2 POSTSCRIPT PRINTER (LN03R)

The other general-use printer supports postscript text and graphics language only. A special print queue form 51 has been set up to support the peculiar aspects of this printer. Only postscript files will print properly on this laser printer. The print command

```
$ PRINT/QUEUE=SYS$LN03R/FORM=51 filename.ext
```

is required to send your output file to this printer.

This printer was upgraded from an LN03-PLUS to support the scientific documentation requirements for text and graphics composition. The WordMARC COMPOSER+ software supports this printer and provides excellent copy. The DISSPLA graphics software is being upgraded currently to also support this printer. (PC users can transfer their postscript files and print them also.)

4.4.3 OTHER NETWORKED PRINTERS

Several other network printers have been added to the Space Systems network. They are for special use applications and systems not located in the vicinity of the general use printers.

Users that have a PC system connect to a terminal server can use a locally-written KERMIT script file to send their PC output file to a variety of laser printers, in particular the LN03+ ascii and LN03R postscript printers. See the network system manager for access to this capability.

5 SOFTWARE DESCRIPTIONS

5.1 DEC SYSTEM SOFTWARE

5.1.1 VMS OPERATING SYSTEM

VMS Version 5.2 is currently used on the MicroVAX II, MicroVAX III, and VAXstation III computers. VMS is now modularized and is designed for all MicroVAX and VAXstation class of computers. The new modularized versions of VMS provide all of the functionality of the full blown VMS operating system and are in every way equal and synonymous to VMS. With the introduction of version 5.0 there was only one VMS operating system for use on all VAX systems; VMS 5.2 operates in the same manner whether you are on a MicroVAX or a VAXstation computer system.

VMS provides scientific users with an efficient working environment and provides many basic software tools to render the hardware useful. A working account of the VMS features can be obtained from the Digital "VMS User's Manual".

Users should become familiar (ASAP) with the online HELP facility; the HELP facility will save time for most users. The Digital Command Language (DCL) command HELP provides current information about the software products installed and a current DCL command syntax. For example, any users logged onto a MicroVAX or VAXstation computer can type the following help command at the system prompt

\$ HELP

to obtain information about the current DCL commands and many other features about that particular system.

VMS system-related software is stored on \$DISK1, \$DISK6, \$DISK11, and \$DISK12; this software is maintained and upgraded periodically.

5.1.2 DECNET-VAX

DECNET-VAX is a VMS optional software layered product that supports the Ethernet hardware. DECNET provides the necessary network communication for file transfer, node-to-node communication, remote node access, etc. We currently have version 5.2 installed as an integral part of the VMS operating system. New updates are added with new versions of VMS automatically.

The DECNET software is also stored on the system disks.

The user-oriented aspects of DECNET are included in the "VMS User's Manual". System aspects of DECNET are included in the "Guide to Networking on VMS." DECNET's operation is generally transparent to most users and is booted up as part of VMS when the VAX systems are powered up or rebooted.

5.1.3 DECSERVER 100/200

The DECSERVER 100/200 version 2.0 software is a layered software product that was installed on the VAXes to support the DECSERVER 100/200 terminal servers. This software is used by the system to configure the network data base for the terminal servers and to downline load the terminal servers when they are powered up. The terminal servers must be downline loaded before they are operational for general use.

This software is generally transparent to most users and requires no special user documentation, inputs, or interaction. A load host must, however, be operational on the Ethernet when the terminal servers are powered up; the server software will not be properly downline loaded if all host computers are inoperable when a terminal server on the Ethernet is turned on. If a server seems to be inoperable, contact the system manager to have the software properly loaded.

The latest version of this software is also stored on the system disks.

5.1.4 LATPLUS

LATPLUS was originally installed on the VAX systems to support remote terminals and printers on the DECSERVER terminal servers. LATPLUS is, however, now included as part of the VMS operating system (version 4.6 or later). LATPLUS provides remote terminal access to the VAX computer systems from Digital terminal servers. LATPLUS runs concurrently with DECNET and also provides remote printer support as described earlier.

LATPLUS is stored on the system disks.

5.2 DEC GENERAL-USE LAYERED PRODUCTS

There is currently only two Digital, general-use, layered software products installed on the VAX computer systems.

5.2.1 VAX FORTRAN

VAX FORTRAN version 5.3 is installed on the systems. VAX FORTRAN is a compiler for scientific software development. Due to popular demand and use of this software product, it is upgraded regularly to provide a current operational version for general use.

This software is stored on the system disks and is available for general use.

To use the FORTRAN compiler just type the FORTRAN command:

```
$ FORTRAN filename.ext
```

Full online help is provided with the DCL Help Facility. To get online help just type the HELP command:

```
$ HELP FORTRAN
```

The Digital documentation "Programming in VAX FORTRAN" provides a full explanation of its use and the general programming environmental aspects such as creating your source files, editing the source files, compiling the source, and linking your object modules.

5.2.2 VAX C

VAX C version 3.1 is installed on the systems. VAX C is a compiler for scientific and system software development. Due to popular demand and use of this software product, it will be upgraded regularly to provide a current operational version for general use.

This software is stored on the system disks and is available for general use.

To use the C compiler just type the CC command:

\$ CC *filename.ext*

Full online help is provided with the DCL Help Facility. To get online help just type the HELP command:

\$ HELP CC

The Digital documentation "Programming in VAX C" provides a full explanation of its use and the general programming environmental aspects such as creating your source files, editing, etc.

5.3 THIRD PARTY SOFTWARE

5.3.1 CALC

CALC is an online calculator that was adapted to the VAX environment. It is designed to calculate arithmetic expressions. In its basic form, expression evaluation is similar to that used by ANSI FORTRAN with calculations performed on INTEGER*4 and REAL*8 constants. Variables may be invoked but are limited to single alphabetic characters. Additional features include octal, hexadecimal, and multiple-precision arithmetic capabilities. Command file control is supported for frequently used expressions.

The software is stored in the temporary calc directory \$DISK3:[GENERAL] and can be executed by typing the CALC symbol that is defined in the SYLOGIN.COM file:

\$ CALC

Documentation and user information for CALC is provided in the CALC appendix at the back of this "Space Systems Network User's Guide." Source code is also provided for those interested in the program's features.

5.3.2 CODEV

CODEV is a complex comprehensive computer program developed for the design and analysis of optical systems. It has become the workhorse for optical design and analysis in the Space Systems Branch at AEDC. Version 7.31A is currently installed on the ELAGPX node, and the software is maintained with a current *single-user license* from the optical design and analysis group Optical Research Associates.

CODEV provides most of the desired capabilities needed to perform a complete design and analysis effort on very simply to very complex optical systems. The software capabilities are summarized in Table 5-1 below.

<u>Some CODEV Capabilities</u>			
Surface Types	System Types	Image	Lens Analysis
Spherical	Objectives	Spot Diagram	Lens Drawing
Cylindrical	Zoom	Optical Path Diff.	Lens Weight & Cost
Aspheric	Anamorphic	Diffraction MTF	Ghost Img./Narcis.
Aspheric Toroid	UV/Visible/IR	Parallax	Element Drawing
Fresnel	Spectrographic	Partial Coherence	Thermal Environ.
Diffraction Grating	Simulators	Spread Function	Pressure Environ.
Tilted	Gradient Index	Knife Edge Scan	Spectral Trans.
Decentered	Multi-Layer	Encircled Energy	Image Simulation
Holographic	Fourier Transform	Gaussian Beams	Relative

Table 5-1. CODEV Optical Des. and Analysis Capabilities.

The CODEV software is stored on the user disk in the directory space ELAGPX\$DISK8:[CODEV].

Users require at least 12000 blocks of pagefile space to run CODEV, and because there is currently only a single user licence, it is not provided for all general users (see the system manager).

Users access CODEV by logging into the ELAGPX node and executing the CV7 DCL command

\$ CV7

Complete documentation supplements the online help features and is available in a Several-volume set. A "CODEV User Information Guide" is included at the end and part of this "Space Systems Network User's Guide". It explains the locally-tailored features (plotters, terminals, etc.).

5.3.3 DISSPLA

DISSPLA is a high-level FORTRAN graphics subroutine library for interactive or batch data representation applications. DISSPLA's integrated subroutines allow programmers to generate graphs, charts, maps, surfaces, contours, and 2- and 3-dimensional designs. DISSPLA's complete functionality, flexibility, and productivity have made it an excellent choice for a wide range of applications. Version 11.0 is currently installed on the VAXes and available for general use.

ALL options are currently licensed and include Shaded Fonts, Page Layout, Business Features, Dynamics, Mapping, Post-Processor, Contouring, and GKS. Users link to DISSPLA using

\$ LINK *filename*, 'DISLINK'

where *filename* is your object mode and *dislink* is a symbol defined by the system. For more information type the HELP command

\$ HELP DISSPLA

Two general documents exist that describe the use of DISSPLA:

- "DISSPLA User's Manual"
- "DISSPLA Pocket Guide"

A FORTRAN subroutine called TPLOT has been written to allow users to quickly use graphics and DISSPLA. The TPLOT appendix better describes the TPLOT subroutine's capabilities.

5.3.4 GUERAP

GUERAP III is a modified and improved version of the original GUERAP program developed by Honeywell Corporation under contract for simulating radiant energy transfer between surfaces of a system. The principle application of the program is to determine an optical sensor's sensitivity to stray energy (radiation) from both external and internal radiation sources.

The program is stored on the system disk \$DISK3 in the directory \$DISK3:[GUERAP].

Users can run the program directly using the DCL RUN command

\$RUN \$DISK3:[GUERAP]GUERAP

The default input and output is SYS\$INPUT & SYS\$OUTPUT, respectively. To read in a data file reassign the logical FOR005 to point to your data file

\$ DEFINE FOR005 *input-filename.ext*

and redirect the output to a file if you do not want it to come to your screen by default (SYS\$OUTPUT equals TT:)

\$DEFINE FOR006 *output-filename.ext*

Use of GUERAP is described in the local TMR "General Unwanted Energy Rejection Analysis Program" (AEDC-TMR-82-V19 by S.L. Steely)

5.3.5 ISIS

The Integrated Simulations for Imaging Systems (ISIS) computer program was developed on and for the VMS operating environment by Applied Technology Associates. The ISIS software package is a modular set of simulation codes which may be selected in many sequences to effect overall electro-optical simulations. The pack-

age includes the generation of time history and image data, an image parameterization code which extracts time history signals from sequences of images, and modules which simulate optical, detector and amplifier signal processing. Utility codes are provided to plot and examine the simulated data.

The software is stored on the user's disk in its own account:

\$DISK4:[ISIS]

Users require at least 35000 blocks of pagefile quota to run the ISIS program. Due to the resources needed, this program has not been set up for general use on the MicroVAX II (see the system manager). To run the program, users need to initialize their workspace and logicals by executing the ISIS SETUP.COM file

\$ @\$DISK4:[ISIS]SETUP.COM

then enter the ISIS symbol to run the ISIS programs.

\$ ISIS

ISIS is documented in two separate binders:

- "Integrated Simulations for Imaging Systems User's Manual"
- "Integrated Simulations for Imaging Systems Program Maintenance Manual"

5.3.6 KERMIT

Kermit-32 version 3.3 has been installed on the VAXes to support file transfers with other local and remote computer systems. Kermit-32 is a program that implements the Kermit file transfer protocol for the Digital Equipment Corporation VAX series of computers under the VAX/VMS operating system.

The software is stored on the system disk \$DISK1 in the systems directory SYS\$SYSTEM and SYS\$HELP.

To run the program, users should enter the RUN command

\$ RUN SYS\$SYSTEM:KERMIT

The program is fully documented in the VMS Kermit appendix at the back of this "Space Systems Network User's Guide".

5.3.7 OPTICAL SIGNATURE CODE

The Optical Signature Code (OSC) version XV is a large computer code used for predicting the radiation received from targets and backgrounds as seen by long-wave infrared sensors in ballistic missile defense scenarios. Reentry vehicles, pen aids, backgrounds, and sensor performance are modeled in exo- and endo-atmospheric environments; the spectral range is from 1 to 30 microns. Several new additions to the OSC have been made to warrant its acquisition and installation at AEDC. OSC Version XV has been adapted to run on the local MicroVAX III computer node FPAMIC. The code is currently operational and installed in the FPAMIC directory \$DISK11:[OSC].

5.3.8 POLYPAGOS

POLYPAGOS, Polychromatic Program for the Analysis of General Optical Systems, uses geometric and frequency response techniques for optical calculations. POLYPAGOS was obtained from The Aerospace Corporation and modified to run on the AEDC CRAY-XMP and the MicroVAX II computer systems.

Four primary tools are provided for measuring optical system performance:

- Calculation of first order parameters and Seidel aberrations coefficients
- Full-field ray trace calculations
- Spot diagrams
- Two-dimensional polychromatic Optical Transfer Function calculations

Both the full-field ray trace and spot diagrams are generated from geometrical aberrations; the Optical Transfer Function measures the frequency response of a system subject to geometrical and diffraction effects. The program produces plots of the spot diagrams, two-dimensional Point Spread Function, and the two-dimensional Modulation Transfer Function.

The source and executable (original and modified) are available and stored in the directory \$DISK3:[POLYPAGOS]

The input and output files correspond to the logical unit numbers 5 and 6, respectively. Users should point FOR005 and FOR006 to their corresponding files (input and output data files).

Several documents prepared by Aerospace Corporation exist that explain how to use POLYPAGOS:

- "POLYPAGOS User's Manual"
- "POLYPAGOS: Polychromatic Program for the Analysis of General Optical System"

5.3.9 SLATEC

SLATEC version 3.2 is a large collection of FORTRAN mathematical subroutines brought together in a joint effort by the Air Force Weapons Laboratory, Lawrence Livermore National Laboratory, Los Alamos National Laboratory, Magnetic Fusion Energy Computing Center, National Bureau of Standards, Sandia National Laboratories (Albuquerque and Livermore), and the Martin Marietta Energy Systems Incorporated at Oak Ridge National Laboratory.

SLATEC is characterized by portability, good numerical technology, good documentation, robustness, and quality assurance. The Library is divided into the following subsections following the lines of the NBS classification system: Elementary and Special Functions, Elementary Vector Operations, Solutions of Systems of Linear Equations, Eigenanalysis, QR Decomposition, Singular Value Decomposition, Interpolation, Solution of Nonlinear Equations, Optimization, Quadrature, Ordinary Differential Equations, Partial Differential Equations, Fast Fourier Transforms, Approximations, Psuedo-random Number Generation, Sorting, Machine Constants, and Diagnostics and Error Handling.

The SLATEC subroutines have been compiled and stored in the library file \$DISK3:[SLATEC]SLATEC.OLB. This library file was produced by compiling all SLATEC source modules and storing them for general use. The file is available for users to link with their programs. For example you link your source object file with the SLATEC library using the LINK command:

```
$ LINK your_object_file.ext, $DISK3:[SLATEC]SLATEC.OLB/L
```

The original source code is also available to users along with a program (\$DISK3:[SLATEC]SLAT.EXE) to extract the desired subroutine(s) and all unresolved externals. The source code is sometimes useful to include in your programs to increase its portability and completeness.

An online HELP program has been modified and stored as \$DISK3:[SLATEC]SLATHELP.EXE.

Both of these programs can be used and are accessible through the command file \$DISK3:[SLATEC]SLATEC.COM generated for general use. After logging into a VAX computer system, users activate this command file by entering the SLATEC system symbol, i.e.

\$ SLATEC

The SLATEC subroutines are categorized and summarized in the attached SLATEC User Information appendix.

5.3.10 TIGER

The Integrated TIGER Series (ITS) version 2.1 is a coupled electron/photon Monte Carlo transport code. ITS is a powerful and user-friendly software package permitting state-of-the-art Monte Carlo solution of linear time-integrated coupled electron/photon radiation transport problems, with or without the presence of macroscopic electric and magnetic fields of arbitrary spatial dependence. The ITS program is fully documented and provided by the Oak Ridge National Laboratory. This code is currently stored and used on the MicroVAX II node MICROS in directory \$DISK5:[TIGER].

5.3.11 VISAGE

The VAX Interactive Signal Analysis and Graphics Environment (VISAGE) was developed jointly by Applied Technology Associates (ATA) and the Air Force Weapons Laboratory (AFWL). The program was acquired from AFWL for use at AEDC.

VISAGE is a single program developed on and for a VAX/VMS using the VAX-FORTRAN compiler. VISAGE's capabilities range

from the simple plotting of time history data, to computing and plotting power spectral densities, to computing coherence and cross spectral densities, as well as many other signal processing operations.

VISAGE has very extensive graphing capabilities. The current version has been modified to support the DISSPLA graphics post-processor and the generation of device independent output (meta files).

The necessary files are stored on the system disk \$DISK3 in the subdirectory \$DISK3:[VISAGE].

Users require at least 30000 blocks of pagefile quota to execute VISAGE and consequently all user accounts are not currently configured to run VISAGE as a result of the resources it requires. Authorized users who run the program need to initialize their workspace by executing the setup command file

```
$ @$DISK3:[VISAGE]SETUP.COM
```

then the correct logicals and the user symbol VISAGE will be defined to allow VISAGE to run properly. After running the setup command file, users execute VISAGE by the VISAGE symbol, i.e.

```
$ VISAGE
```

Use of VISAGE is fully documented in the "VISAGE User's Guide."

5.3.12 WORDMARC COMPOSER +

COMPOSER+ is the latest version of the scientific wordprocess produced by MARC software. COMPOSER+ is designed to run on many systems (including PCs) and was recently installed and activated for general use on the MicroVAX II, as of this writing.

COMPOSER+ supports mathematical copy and graphics composition. It is especially useful for scientific reporting and general documentation preparation. Current use includes excellent generation of presentation quality viewgraphs.

New users are added to a COMPOSER+ terminal definition file and can execute the COMPOSER+ software by entering the WM command

\$ WM

A full description of the available documentation and locally-added features and the hardware provided is supplied in the COMPOSER+ appendix at the end of this "Space Systems Network User's Guide".

Space Systems Network User's Guide

Appendices

CALC User Information

CODEV User Information

COMPOSER + User Information

SLATEC 2.0 User Information

TPLOT User Information

VMS KERMIT User Information

CALC

USER INFORMATION

(This appendix was modified from a file provided with the **CALC** program. No attempt was made to edit or modify the file other than to format it for inclusion in the **MicroVAX II User's Guide** as a result of general user interest.)

CALC is a calculator designed to evaluate arithmetic expressions. In its basic form, expression evaluation is similar to that used by ANSI FORTRAN with calculations performed on INTEGER*4 and REAL*8 constants. Variables may also be invoked but are limited to single alphabetic characters. It is assumed that the reader is familiar with FORTRAN data types, constants, expression syntax, operator precedence, and the syntax for assigning values to variables. Additional features include octal, hexadecimal, and multiple precision arithmetic capabilities. Commonly used commands and expressions can be placed in a file and executed when convenient.

-GETTING STARTED-

To run **CALC** on your system, you must initiate the commands appropriate for your particular operating system. Contact your system manager for specific details. Once running, **CALC** prompts for input with

CALC>

Try typing

123+456

followed by a carriage return. **CALC** will evaluate the expression and output the answer

579

It then prompts for further input. Try other expressions such as

12.0 - 99.	(answer=-87.00000000000000)
-(-32767+(6-2)**8-(512/(409-401)))	(answer=-32705)
3*5/7	(answer=2)
3*(5/7)	(answer=0)

Mixed mode is legal, for example

1977/50. is evaluated as 39.54000000000000

Reals may be expressed using D or E format. For example

1.2E10*2.D0**3-1.D-8 is evaluated as

0.95999999999999992D+11

Variables may also be used to retain values for later use. CALC allows variables consisting of a single alphabetic character. As in FORTRAN, variable A through H and O thru Z default to type real, I thru N to type integer. To set I to a value use the usual FORTRAN syntax, for example:

I=2**10-1

Try typing the single character 'I'. CALC will respond with its value. We can now use I in various expressions such as

J=I-I/3*3

% is a special variable that retains the value of the last expression evaluated. For example, to successively add up the numbers 1, 2, 3, 4, 5, and 6 we could enter

1
%+2
%+3
%+4
%+5
%+6

Note that you can examine the value of the variables by typing the appropriate single character followed by a carriage return. Such an examination does not change the value of %.

To exit from CALC, type

*E (or *EXIT)
or *S (or *STOP)

-SPECIAL FUNCTIONS-

CALC recognizes a variety of special functions. For example, to calculate the square root of 2, we can type

SQRT(2.)

CALC responds with the value 1.41421356237310

Each function may have an expression for its argument. For example,

A=2.0*SQRT(ALOG(9.)+3.)

sets A to 4.55948443459838

The following special functions are available:

<u>FUNCTION</u>	<u>ARGUMENT</u>	<u>FUNCTION VALUE</u>	<u>DESCRIPTION</u>
ABS	REAL	REAL	absolute value
DABS	REAL	REAL	absolute value
IABS	INTEGER	INTEGER	absolute value
COS	REAL	REAL	trigonometric cosine
DCOS	REAL	REAL	trigonometric cosine
EXP	REAL	REAL	e**X
DEXP	REAL	REAL	e**X
IFIX	REAL	INTEGER	REAL to INT. conver.
AINT	REAL	REAL	REAL truncation
INT	REAL	INTEGER	REAL to INT. conver.
IDINT	REAL	INTEGER	REAL to INT. conver.
ALOG	REAL	REAL	natural logarithm
DLOG	REAL	REAL	natural logarithm
ALOG10	REAL	REAL	logarithm base 10
DLOG10	REAL	REAL	logarithm base 10
SIN	REAL	REAL	trigonometric sine
DSIN	REAL	REAL	trigonometric sine
SQRT	REAL	REAL	square root
DSQRT	REAL	REAL	square root
ATAN	REAL	REAL	arc tangent
DATAN	REAL	REAL	arc tangent
TANH	REAL	REAL	hyperbolic tangent
DTANH	REAL	REAL	hyperbolic tangent

-WORKING IN OCTAL AND HEXADECIMAL-

You may change the base used to specify constants by using the *B command. Legal forms are

command	action
-----	-----
*B	displays current default base
*B 8	changes default base to octal
*B 10	changes default base to 10
*B 16	changes default base to 16

Suppose we have changed the default base to octal. Then adding

$$7 + 1$$

we obtain the result

00000000010 (BASE 8)

If the default base is hexadecimal, we can enter

$$9 + 1$$

which is evaluated as

0000000A (BASE 16)

Suppose we have assigned

$$A=1$$

then

$$1+A$$

gives

2.0000000000000000

even when the default base is 16. If we wish to add the hexadecimal digit 'A' to 1, enter

1+0A

We now obtain the desired

0000000B (BASE 16)

This leading 0 is only necessary when the first hexadecimal digit is greater than 9.

If constants are entered with digits that are not legal for the base being used, the entire number is converted using a more appropriate base. For example, if we have set the default base to octal and type

1+9

the 9 is not an octal number so it is converted to base 10. If a base 16 number is involved, the result will be in base 16.

You may temporarily change the base for a single integer constant by preceding it with

$\wedge 8$	for octal
$\wedge 10$	for base 10
$\wedge 16$	for base 16

For example, if the default base is 10,

100+ $\wedge 8$ 40

gives

132

a base 10 integer.

I=100+ $\wedge 16$ 10

gives

116

also a base 10 integer.

Note that the '^' can only be used to specify the base of constants and that expressions such as ^16I are illegal.

To declare variables to be integers of a specific base, we can use the commands

*INTEGER	(base 10)
*OCTAL	(base 8)
*HEX	(base 16)

for example,

*INTEGER A	declares variable A to be a base 10 integer.
*HEX B,Z,F	declares variables B, Z, and F to be base 16 integers.
*DECIMAL	lists all the variables that have been declared to be of type DECIMAL.

To summarize, there are three distinct ways of making base declarations when using CALC. The first is to use the *B command to designate the base default value. This is used to determine the base for constants when they occur in expressions. It does not in any way influence the type of any variables found in an expression. The only way to change the type of a variable is with a specific CALC command such as

***INTEGER A,B**

Suppose for example that the default base is 10 and we enter

***OCTAL A**
A=100

then CALC responds with

00000000144 (BASE 8)

Finally, the last way to change a base is to use the explicit base specifiers for a constant, for example

`^10 123`

`^8 777`

`^16 AB`

-MULTIPLE PRECISION-

Normally integer arithmetic (base 8, 10, and 16) is done internally with INTEGER*4 variables. To allow for larger numbers, CALC has multiple precision capabilities that allow numbers up to 99 digits to be manipulated. Constants are converted to a multiple precision data type when the number of digits specified exceeds a certain value. This value depends upon the specified base. Leading zeroes are included in this count and can be used to force constants to be of type multiple precision.

<u>base</u>	<u>maximum number of digits before conversion</u>
8	10
10	9
16	7

Suppose we type (with the default base of 10) the number

1234567890

then CALC echoes with

1,234,567,890
(BASE 10)

The commas indicate that % now has type multiple precision base 10. Similarly, typing

1234ABCD

results in

1234,ABCD
(BASE 16)

Notice that base 16 multiple precision numbers are separated by commas every 4 digits, octal and base 10 numbers every 3 digits.

You may perform the usual operations of addition, subtraction, multiplication, division, and exponentiation. As of version 1.0,

exponentiation of a multiple precision number may only be to a non-negative integral power. To declare variables of type multiple precision, use

*M8	(multiple precision base 8)
*M10	(multiple precision base 10)
*M16	(multiple precision base 16)

for example,

***M8 A,B** declares A and B to be multiple precision octal variables.

Then typing

A=32768

results in CALC responding with

100,000
(BASE 8)

-ADDITIONAL COMMANDS-

All commands to CALC (as distinguished from expressions to be evaluated) begin with an asterisk. To obtain a list of all possible commands, type a question mark followed by a carriage return. Most of the commands have already been described. The following section gives an explanation of the remaining commands.

<u>COMMAND</u>	<u>DESCRIPTION</u>
*@filename	Where filename is the name of a file of CALC commands. CALC reads the file and executes the commands. Up to 5 nested calls can be made. Recursive calls are not allowed. CALC prompts with CALC<n> before each command line is executed, where n is the calling level. You may optionally follow the file name with a blank followed by a single variable name (a single alphabetic character or %). CALC will then execute the file until the value of that variable is zero or negative. The test of this variable is made before the file is executed and not during execution of commands within the file. If the variable's value is not positive when the command is initially encountered, the file will not be opened for execution. See the section on command file examples for ways to use this option.
*ASCII	Declares a list of variables to be of type ASCII. Useful when decoding ASCII characters. For example, if we set A to be of type ASCII, then typing A=77 results in the character 'M' being output. The inverse operation is the single quote. It allows us to specify a single ASCII constant. For example, if we type 'M then the character 'M' is echoed and indicates that % holds that character and has data type ASCII. Suppose that the variable I has data type INTEGER. Then we can output the base 10 code for the ASCII character 'M' by

entering I='M which results in 77 being output. Notice that you may not be able to enter certain control characters that are intercepted by your operating system. Characters whose value is less than 32 are output by printing the character '^' followed by the equivalent ASCII character of that number plus 32. For example, A=10 results in ^* being output since 42 is the ASCII code for the character '*'. See appendix A for a table of the characters output by CALC to represent such non-printable characters.

***C** COMMENT line. The characters that follow are ignored by CALC. This is useful when documenting files containing CALC commands.

***N** NOVIEW. Prevents CALC from outputting the value of the expressions evaluated. This is especially useful when executing files containing CALC commands that initialize variables to special values. Equivalent to *V 1

***V** VIEW. Controls CALC's printing options:

<u>command</u>	<u>output class</u>
*V 0	error messages
*V 1	error messages command lines read from files
*V 2	error messages value of expressions evaluated
*V 3	error messages command lines read from a file value of expressions evaluated
*V	same as *V 3

The default setting is *V 3. Notice that other legal forms are *VIEW 1 and *V2

- *R** **READ.** Allows a single line to be read from the terminal. Useful in files of CALC commands to allow additional commands to be entered (like *S to exit from that file) or simply as a way to halt terminal output until the carriage return key is pressed.
- *REAL** Declares specified variables to be REAL*8. When the value of such variables are output, FORTRAN's D format is used.
- *DECIMAL** Declares specified variables to be REAL*8. When the value of such variables are output, FORTRAN's F format is used. Variables A-H and O-Z default to type DECIMAL.
- *S** **STOP.** Same as *E
- *E** **EXIT.** Terminates CALC session unless it is used within a file of CALC commands. In this case, CALC closes the file and continues with the next command.
- *Z** **ZERO.** Zeroes all variables except %. Data types are not changed.

-ADDITIONAL FEATURES-

CALC is similar to **FORTRAN** with respect to operator precedence. Blanks may occur anywhere on a command line without effect except after a single quote mark used to specify a single ASCII character constant. **CALC** extends the ANSI **FORTRAN** syntax by allowing the following:

1. multiple assignments on one line, for example

I=J=K=812

2. Unary + and unary - are allowed, for example

2*-3

+2+-7

-24**

are all legal. The last expression evaluates to 16 because the unary - has a higher precedence than exponentiation.

3. Exponentiation may be indicated by using ! as well as **

If any of the declarations are entered (such as ***INTEGER** or ***M8**) and no argument to this command is given, then **CALC** will print out the variables that have been assigned that data type. Note that a variable can be assigned to different data types using such commands and still not be assigned a value. If you attempt to output the value of such a variable, an error message will result.

-COMMAND FILE EXAMPLES

EXAMPLE 1:

PROBLEM: be able to enter the coefficients of a second degree polynomial and have the roots output.

Solution: create the following file and call it ROOT:

```
*CALCULATES THE ROOTS OF A 2ND DEGREE EQUATION
*C YOU WILL BE ASKED TO ENTER THE VALUES OF
*C A,B AND C WHERE
*C
*C 2
*C A X +B X + C = 0
*C
*C
*C ENTER THE VALUE OF A
*R
A=%
*C ENTER THE VALUE OF B
*R
B=%
*C ENTER THE VALUE OF C
*R
C=%
*C THE ROOTS ARE:
X=(-B+SQRT(B*B-4.*A*C))/2.*A
Y=(-B-SQRT(B*B-4.*A*C))/2.*A
*C
*C AS YOU CAN SEE BECAUSE
A*X*X+B*X+C
A*Y*Y+B*Y+C
```

Then run the procedure by entering CALC and typing

***@ROOT**

EXAMPLE 2:

PROBLEM: Suppose we are working on a problem that requires us to convert many decimal 16 bit word addresses to octal byte address. We would like to be able to simply enter the decimal word address and have CALC respond with the octal byte address.

SOLUTION: Create a file of the following commands and call it BYT:

```
*C
*C
*C
*C
*C ENTER DECIMAL WORDS
*R
A=%+%
```

Then we enter CALC and type the following:

```
*OCTAL A      (to make A an octal variable)
1             (to set % to a non-zero number)
*@BYT %       (to execute the file of conversion commands)
```

We will then repeatedly execute the file BYT until we enter a zero as the number to be converted.

EXAMPLE 3:

Problem: We wish to set up a command file called SIGN that allows us to enter a number, and then execute the files MINUS, ZERO, and PLUS according to the value entered.

Solution: Create the file SIGN consisting of:

```
*C ENTER THE NUMBER WHOSE SIGN IS TO BE DETERMINED
*R
I=J=%
I=-I
K=1
*C K STAYS AT 1 IF NUMBER IS NON-ZERO
*@MINUS I
*@PLUS J
*@ZERO K
```

The file MINUS:

```
*C THE NUMBER IS NEGATIVE
I=K=0
```

The file PLUS:

```
*C THE NUMBER IS POSITIVE
J=K=0
```

and the file ZERO:

```
*C THE NUMBER IS ZERO
K=0
```

Notice that K is used to control the execution of the file ZERO by initializing it to 1 and resetting it to zero if either of the files PLUS or MINUS are executed.

EXAMPLE 4:

PROBLEM: Enter the number N and calculate the numbers
1!, 2!, 3!, 4!, ... N!
where
$$N! = N*(N-1)*(N-2)*...*3*2*1$$

SOLUTION: Since the numbers may grow very large, we will use a multiple precision variable. Create the following files:

MFACT:

```
*C INPUT THE LIMIT OF THE FACTORIAL LIST
*V 0
*R
N=%
*M10 A
I=0
*@MF1 N
*C SET VIEW AND TYPE FOR VARIABLE A AT DEFAULT VALUES
*DECIMAL A
*V
```

MF1:

```
A=1
N=N-1
I=I+1
J=I
*@MF2 J
*V2
A
*V0
```

MF2:

**A=A*J
J=J-1**

We then enter CALC and type

***@MFACT**

A typical run of this procedure would look like:

```
>CAL  
CALC>*@MFACT  
CALC<2>*C INPUT THE LIMIT OF THE FACTORIAL LIST  
CALC<2>*V 0  
CALC<2>4  
1  
(BASE 10)  
2  
(BASE 10)  
6  
(BASE 10)  
24  
(BASE 10)  
CALC>
```

EXAMPLE 5:

PROBLEM: We wish to specify a default base and successively add numbers into a sum.

SOLUTION: Create the following files of CALC commands:

ADD:

```
*V
*C ENTER DEFAULT BASE
*R
*C CHANGE DATA TYPE OF SUM
*C BY SPECIFYING THE DATA TYPE FOR I AND J
*R
*C TO EXIT, TYPE 0 TO ZERO OUT, ADD -I
*V0
I=0
1
*@ADD1 %
*V
```

ADD1:

```
*V
*R
*V0
J=%
I=I+%
*V2
I
*V0
%=IABS(J)
```

Then we invoke the procedure by typing

```
*@ADD
```

A typical run follows:

```
CALC>*@ADD
CALC<2>*V
CALC<2>*C ENTER DEFAULT BASE
CALC<2>*R
CALC<2>*B 8
DEFAULT BASE IS 8
CALC<2>*C CHANGE DATA TYPE OF SUM
CALC<2>*C BY SPECIFYING THE DATA TYPE FOR I AND J
CALC<2>*R
CALC<2>*OCTAL I,J
CALC<2>*C TO EXIT, TYPE 0  TO ZERO OUT, ADD -I
CALC<2>*V0
CALC<3>177
00000000177 (BASE 8)
CALC<3>1
00000000200 (BASE 8)
CALC<3>0
00000000200 (BASE 8)
CALC>
```

USAGE NOTES VERSION 1.0

1. When you iterate on a file by a call such as

***@REPEAT X**

then note that

- A) X must have been set to a positive value when the command is executed or else the file will not be executed.**
 - B) If the file of commands does not change the value of the variable X you will enter an infinite loop. You can explicitly set X to a non-positive value, use the *Z command to zero it (if it is not %), or include a *R command to give you a chance to reset the variable and get out of the loop.**
 - C) *E and *S will allow you to exit from the command file REPEAT but will not of themselves prevent repetitions.**
 - D) Entering constants echo on the terminal (assuming *V is properly set) and can change the value and type of the variable %. This is important to remember when using % to control the iteration of a file.**
-
- ### **2. When you first enter CALC, the variable % has type INTEGER and holds the version number of the program.**
-
- ### **3. In practice, multiple precision arithmetic may be limited to less than 99 digits because of your terminal's inability to print that many characters.**
-
- ### **4. No implicit conversion is made to multiple precision when operations with reals or integers cause an overflow. This was done in version 1.0 in case the multiple precision routines have to be removed when creating a small task image for some operating system.**

5. In FORTRAN,

$-A^{**2}$ is the same as $-(A^{**2})$

with CALC,

$-A^{**2}$ is the same as $(-A)^{**2}$ (just like SNOBOL!)

6. If R and A are positive reals and I is a positive integer, some compilers like RSX-11M's F4P won't allow (at run time) evaluation of

$(-I)^{**R}$

$(-A)^{**R}$

7. Under RSX-11M F4P you will find

$2^{**.5}$ to have value 1

while

$2^{**.5}$ has value 1.41421356237310

8. 10E10 is a hexadecimal constant (integer) while 10.E10 is a real.

ASCII/CALC CODES

<u>ASCII CODE</u>	<u>CALC PRINTS</u>	<u>ASCII CODES</u>	<u>CALC PRINTS</u>	<u>ASCII CODE</u>	<u>CALC PRINTS</u>	<u>ASCII CODE</u>	<u>CALC PRINTS</u>
0	^	32	64	@	96	'	
1	^	33		65	A	97	a
2	^"	34	"	66	B	98	b
3	^#	35	#	67	C	99	c
4	^\$	36	\$	68	D	100	d
5	^%	37	%	69	E	101	e
6	^&	38	&	70	F	102	f
7	^'	39	'	71	G	103	g
8	^(	40	(	72	H	104	h
9	)	41	)	73	I	105	i
10	^*	42	*	74	J	106	j
11	^+	43	+	75	K	107	k
12	^,	44	,	76	L	108	l
13	^-	45	-	77	M	109	m
14	^.	46	.	78	N	110	n
15	^/	47	/	79	O	111	o
16	^0	48	0	80	P	112	p
17	^1	49	1	81	Q	113	q
18	^2	50	2	82	R	114	r
19	^3	51	3	83	S	115	s
20	^4	52	4	84	T	116	t
21	^5	53	5	85	U	117	u
22	^6	54	6	86	V	118	v
23	^7	55	7	87	W	119	w
24	^8	56	8	88	X	120	x
25	^9	57	9	89	Y	121	y
26	^:	58	:	90	Z	122	Z
27	^;	59	;	91	[	123	{
28	^<	60	<	92	\	124	
29	^=	61	=	93	]	125	}
30	^>	62	>	94	^	126	~
31	^?	63	?	95	_	127	

CODEV

USER INFORMATION

USING CODEV IN THE VMS ENVIRONMENT

CODEV has been installed on the VAXstation III computer node ELAGPX. The following instructions are mainly concerned with using the optical analysis program CODEV (current Version 7.31) in the VMS environment. The input, output, starting, stopping, files, etc. are of primary interest for the following instructions. The technical issues are covered in a formal set of CODEV documentation; users should refer to the CODEV Prompting Guide or the CODEV Reference Manual for more details on using CODEV and the individual CODEV commands.

1.0 Signing on to CODEV

In response to the MICROVMS prompt, \$, type in either

CV7	- responds with WELCOME screen - enters screen mode; go to command mode anytime by first pressing the PF1 key (called the "GOLD" key) and the C (for "Command")
------------	---

or type

CV7/qualifiers

where valid /qualifiers are:

/COMMAND	- enters CODEV in command mode
/RECOVER=recfile	- users recfile.REC as input file; CODEV.REC is the default
/DEFAULT=deffile	- uses deffile.SEQ instead of DEFAULTS.SEQ
/BATCH=filespec	- submit run defined in filespec (default type .SEQ) to the queue defined at the site as SYS\$BATCH

/QUEUE=quename

- changes queue of /BATCH command to quename

/AFTER=vax_timespec

- submit batch run after given date and time

Qualifiers can be entered in any order; each qualifier needs only enough letters to make it identifiable from the others -- 1 letter in all these cases.

Special relationships:

/BATCH and /RECOVER can only be run in command mode, so /COMMAND isn't a necessary qualifier

/QUEUE and /AFTER only apply to /BATCH runs

Examples:

Enter in command mode for interactive processing:

CV7/C

Process default recover file (CODEV.REC) in command mode:

CV7/R

Deferred batch with a special defaults file:

CV7/A=15-AUG-1988:20_00/D=PROJECT/B=DBGAUSS

Puts run on SYS\$BATCH queue, deferred to start after 8:00 P.M. on August 15th, 1988; it will set up the defaults from the file PROJECT.SEQ and run with the input file DBGAUSS.SEQ. Since these commands are being interpreted by VMS, filename syntax is for VMS, not CODEV; use, if full specifications is needed:

DBGAUSS.SEQ;5 instead of DBGAUSS.SEG(5)

2.0 Exit from CODEV

When you are through with runs on your current lens, you can start work on another without leaving CODEV. When you are all through with your CODEV session, type

EXIT

In screens, exit by typing GOLD/E. You will be asked to confirm exit (answer Y or N in response to the prompt). Control will be turned back to the VAX VMS operating system. If you are using multi-processing (enabled by an initial MPX command) EXIT will only be allowed after all processes are finished.

3.0 Immediate Commands

Most CODEV commands must be given in the LDM or the appropriate option; a few can be given anytime and are called 'immediate commands' because they are recognized when ever input is expected. They fall into 3 categories:

User Helps:

HEL, CPU, SYS, EDI, SCR.

Optical Configuring:

RDM, DDM, DIN, HEA

I/O Configuring:

EJE, CRT, ECH, COL, DEV, IN, OUT, GRA, PRT, PLT

For example:

HEL	- takes you into the help system
SCR	- command puts the program into screen mode
RDM	- set input & output to radius mode rather than curvature
DDM	- allows specification of inches, centimeters, or millimeters as the default dimensional system for lens inputs without any other specification

EJE	- allows inclusion of top-of-form ejects in listed output
COL	- allows setting of default plotting color for line drawings
DEV	- allows designation of default printer, plotter, or terminal type; for example, DEV TER VT240 for terminal.
IN	- assigns input to a file (usually .SEQ type) instead of the terminal for the next series of commands
OUT	- assigns listed output to a file (usually .LIS type); listed output can also be temporarily turned on and off with OUT Yes/No
GRA	- assigns graphical output to a file (usually .PLT type)
PRT	- prints a designated file, using the print device specified by DEV
PLT	- plots a designated file, using the plot device specified by DEV

I/O and file commands will be discussed in detail shortly; see the Prompting Guide for details on the other commands.

Immediate commands can be entered at any time from command mode. When in screens, the quickest way to enter an immediate command is to type GOLD/Q (for quick command entry), then enter the command. Most immediate commands are accessible through the "Defaults" screen under "Defaults & Utilities" (Item 9 on the main menu).

4.0 PROMPTS

The prompts given in CODEV command mode tell you where you are working:

CODEV>	- in the LDM
XYZ>	- in option XYZ, where XYZ is the 3-letter command given to call the option

In screens mode, the identifier in the upper left hand corner includes, as the first 3 letters:

LEN - in the LDM

XYZ - in option XYZ

5.0 OPTION PROCESSING

After completion of lens entry, etc. in the LDM, you enter an option by giving the command for it, or by selecting the option from a menu in screens mode. It then waits for further inputs local to that option. As soon as you complete your entries, start calculation by giving:

GO - in command mode;

GOLD/G - in screens mode.

When the calculation is complete, the program immediately returns to the LDM. In a batch run, a GO must follow an option and its inputs before any LDM commands can be entered.

If, in the process of making entries to that option, you realize you made a mistake in a prior command, you almost always can reenter the command to override the previous entry; the only exceptions to the override of a previous entry are when information is additive, such as when more than one command of a type can be issued (e.g., RSI commands in LAYOUT to draw user-specified rays). If you change your mind and don't want to run the option at all, type

CAN - in command mode;

GOLD/M - in screens mode.

and you will be returned to the LDM for entering data or another option command.

In addition, you can interrupt many options while they are processing by typing CONTROL-C. The option will stop when it reaches the next test for CONTROL-C (this may take several seconds). Not all options test for CONTROL-C, but compute-intensive ones generally do.

6.0 FILES USED BY CODEV

CODEV uses VAX files for a variety of purposes: data input, lens library storage, text and graphic output, saving screen data, and several others. Each of these file types has a unique default extender that identifies its type.

filename.LEN Lens library file; binary file (not editable, can only be written/read by CODEV). RES and SAV commands affect these files, as do commands in the LIB option.

filename.MUL Multilayer coating library file (MUL option); binary file (non-editable).

filename.LIS Text output file (editable).

filename.PLT Plotted output file (editable); also known as "neutral picture file."

filename.REC Recover file; used to recover from aborted run (editable text file similar to .SEQ files).

filename.SEQ Text input file (editable; can contain any CODEV commands).

filename.xxx Screen file for option xxx (where xxx is a CODEV option name -- AUT, TOR, MTF, etc.). Saves user's screen inputs for future use (not editable; can only be written/read by CODEV).

7.0 EDITING FILES

The immediate command EDI calls up the VAX/VMS system default editor (usually EDT; check with your system manager). You can use this to create or modify input files (.SEQ files) or to edit any other type of text files. Please see the appropriate DEC VAX/VMS documentation for instructions on using EDT or other VAX/VMS editors.

Examples:

EDI Newlens - edit Newlens.SEQ

EDI MFTOUT.LIS - edit output listing

8.0 FILE INPUT/OUTPUT, PRINTING, AND PLOTTING

By default (unless changed by DEFAULTS.SEQ file -- see Section 13), input and output are associated with your terminal. CODEV has commands for directing input and output elsewhere (to/from other devices and files). These commands make use of the files discussed in the previous sections.

I/O, printing, and plotting commands are all immediate commands. In command mode, enter them at any time. In screens, enter GOLD/Q for quick entry of a single command, or use the "Defaults" screen under "Defaults & Utilities" (Item 9 on the main menu).

8.1 Input

The command **IN filename** tells CODEV to get input from the file filename.SEQ (you must have previously created filename.SEQ with an editor).

Example:

IN MYLENS1

reads the contents of MYLENS1.SEQ and then returns control to the terminal (or the calling .SEQ file).

8.2 Text Output

Text output can go to your terminal, a file, or both. The command **OUT T filename** directs output to both the terminal and to the file filename.LIS. You can omit either the T or the filename if you want output to a file only or the terminal only.

Examples:

OUT T - output to terminal only

OUT XYZ - output to XYZ.LIS only

OUT T ABC	- output to terminal and to ABC.LIS
OUT T *	- output to terminal and last specified text output file

Note: You cannot use an output filename of T.LIS.

8.3 SEQ File Output (for lenses only)

The command **WRI filename** (Write) will produce a text file **filename.SEQ** containing only the lens data for the lens currently in memory. You can then edit this file or read it into CODEV with **IN filename** (note that this is not the same as the command **SAV filename** which produces the non-editable lens library file **filename.LEN**).

8.4 Writing Version 6 DATA

There is another command, **WRD filename** (for WRite Data) that will output the lens in a .SEQ file using the Version 6 "DATA" option format. This permits you to move a lens from Version 7 to Version 6.95 if this should be necessary. Note that some lens data features of Version 7 (such as tolerances) were not included in the DATA option and are thus not included. Check the resulting DATA file carefully before use. Version 7 can also read DATA option files (this is only a temporary early-version compatibility feature).

8.5 Graphics Output

Graphics output can go to your terminal, a file, or both. The command **GRA T filename** directs output to both the terminal (it must be a graphics terminal; see "Graphics Devices") and the file **filename.PLT**. You can omit either the T or the filename.

Examples:

GRA T	- graphics to terminal
GRA ABC	- graphics data to file ABC.PLT

GRA T XYZ

- graphics to terminal and to XYZ.PLT

GRA T *

- graphics to terminal and graphics data to last specified file

8.6 Graphics Devices

Graphics data is saved in files called "neutral plot files" (.PLT files) or "neutral picture files" (.RAS files). CODEV or a stand-alone "driver" program can translate this data to a form needed by a particular device (graphics terminal or plotter). You must first tell CODEV the type of device by entering a DEV command. The device name you enter must be one of ORA's supported types or one that has been created at your site (check with your VAX system manager).

Examples:

DEV TER VT240

- terminal is a DEC VT240 graphics terminal

DEV PLT PSLASER

- plotter on system is a postscript printer

DEV PLT HP7550

- plotter on system is a 8-PEN HP7550

Note: If you enter a graphics command before defining a device, CODEV will prompt you with a list of supported devices to choose from.

8.7 Sending Data to Printers, Plotters, and Raster Graphics Devices

You can send text output to a printer and graphics output to a plotter or raster device from within CODEV. The data must have been saved first (using OUT[T] filename or GRA[T] filename). You should have also defined your output devices with a DEV command; since this information seldom changes, it is convenient to define it in your DEFAULTS.SEQ file for automatic execution whenever you run CODEV.

Examples:

PRT

- prints current output file on defined printer

- PRT ABC** - prints ABC.LIS on printer
- PLT** - plots current graphics file on defined graphics plotter
- RAS** - sends current graphics file to defined raster graphics device (we currently have none)

9.0 PLOTTING AND PRINTING AFTER EXIT

You can use the normal VMS Print command (see VAX installation guide or VMS manuals):

PRINT/qualifiers filespec

An analogous plot command has also been added:

CVPLOT[/DEVICE=plot_device] [filespec]

where:

- filespecs** - is VMS form (.PLT is default type; CODEV.PLT is default filespec
- plot_device** - are any of your site dependent device names (the current default plotter is an LN03R postscript printer and is in room ELA-311).

10.0 RECOVERY

Whether you are using screens or commands, CODEV always creates a command-mode record of your session input in a "recover file" called CODEV.REC. This file can be used to recreate your interactive session should this be necessary (due to a system crash during your session, for example). Only ONE version is kept; running CODEV again will wipe out CODEV.REC unless you run with the /RECOVER qualifier, i.e.,

\$ CV7/REC

In this case, CODEV will read and execute the contents of CODEV.REC to recreate your last session. You can also edit the CODEV.REC file to alter the course of the "recovery" run or rename the file to a .SEQ type to save it for later use.

11.0 STARTING OTHER PROCESSES FROM WITHIN CODEV

You can run any VMS command from within CODEV by using the SYS command. This puts you temporarily into a VMS "subprocess." Use this to copy files, read your VAX mail, etc.

Examples:

SYS 'MAIL'	- go into VMS MAIL utility
SYS 'DIR *.PLT'	- check to see what plot files exist
SYS	- drops you into subprocess (you must LOGOUT of subprocess to return to CODEV)

12.0 DEFAULTS SETUP

You can tailor your own operating environment because the DEFAULTS.SEQ file is read in before any other inputs. Most of the immediate commands are eligible; these are marked in the Prompting Guide. In addition, the RES (restore command) is also eligible and can be used to read in a lens.

A typical DEFAULTS.SEQ would be:

```
HEA ' OPTICAL RESEARCH ASSOCIATES'
DIN 'ORA'
DDM M
IMPX      (multi-process commented out)
RDM yes
DEV TER VT240
DEV PLT ZETA8
EJE no
ECH no
CRT yes
```

13.0 USER DEFINED FUNCTIONS AND USER VERSIONS OF CODEV

CODEV provides two user defined functions that are of interest to the optical designer. Sample source code with instructional comments is contained in

- | | |
|--------------------|---|
| GRUDEV.FOR | - defines a gradient index function of (x,y,z).
The sample files defines a Luneberg lens distribution. |
| USERSUR.FOR | - defines a function of (x,y,z) describing a user-defined surface (and optionally the functions's derivative). The sample file defines a Zernike-like polynomial surface of 18 terms (and optional conic base surface). |

These may be modified and linked into CODEV by the system manager and thus apply to all CODEV users on the system. Alternatively, a user can make his own customized version of CODEV without changing it for the rest of the users.

Creating a customized version of CODEV requires several steps. The user starts by copying the LINKC.COM procedure and all the .OBJ files from CV7_EXEC. He then compiles the module he has modified to create a .OBJ file for it. Running the LINKC.COM procedure creates an executable CODEV that is linked to the common sharable image CV7SHR.

For instance, if a user wants to define his own surface and link a version of CODEV on the directory DUA0:[USER]:

```
$ COPY CV7_EXEC:LINKC.COM,*.OBJ,USERSUR.FOR DUA0:[USER]
$ EDIT USERSUR.FOR
$ FORTRAN USERSUR
$ @LINKC
```

To run his version of CODEV, the user redefines the logical CV7_EXEC to point to the directory containing his CODEV.EXE. Then he runs CODEV as usual.

```
$ DEFINE CV7_EXEC DUA0:[USER]
$ CV7
```

Note that if the user wants to plot (either with the PLOT or the PLT commands) he must copy the file PLOT.EXE from CV7_EXEC as well as the plot driver files. The following example would allow him to access a Zeta plotter:

```
$ COPY CV7_EXEC:PLOT.EXE,ZETA.COM,ZETA.EXE DUA0:[USER]
```

14.0 FILE CONVERSION UTILITIES (VERSION 6 to VERSION 7)

Two utilities are provided for converting most Version 6 lens libraries (LENS.CAT) into Version 7 lens files (filename.LEN), and for converting Version 6 sequence files into Version 7 sequence files (both with extension .SEQ).

The operation of these utilities is described in the "CODEV Conversion Guide (from Version 6 to Version 7)," Chapter V. However, the command names have been changes since printing the Conversion Guide (an errata sheet for that manual will also point this out).

Command (from VMX \$ prompt):

\$ CVCAT	- instead of CONVCAT for lens libraries
\$ CVSEQ filename	- instead of CONVSEQ for .SEQ files.

15.0 CODEV Accounting

The CODEV accounting file contains the cpu time used and date/time stamp of each CODEV access. This file is initialized at the time CODEV is first installed on our system. CODEV will not run unless this file is present and intact. This is true for all forms of CODEV licensing, although only variable fee licensees are required to produce and send monthly reports to ORA. At our site, Jim Selman is designated to take care of accounting requirements, so most users will not be aware of them.

**COMPOSER +
USER INFORMATION
PRINTERS
TYPESTYLES
AND
THE MICROVAX II**

COMPOSER+

The MicroVAX II users have a new scientific documentation tool at their disposal. The software package WORDMARC COMPOSER+ provides many new features that the old WORDMARC COMPOSER did not provide. Use of the new COMPOSER+ is similar to the old COMPOSER. To use the new COMPOSER+ software you have to be sure you have defined the logical WM\$IDENT for your session so COMPOSER+ knows your terminal location and type. To assign the logical just type the DCL command.

```
$ ASSIGN username WM$IDENT
```

where 'username' is your login username. This assignment sets up a logical pointer to your terminal to accommodate the dynamic network logical terminal assignment when you login to the system. You may want to put this command in your LOGIN.COM command file. As an example the user SMITH would have the following command in his LOGIN.COM command file.

```
$ ASSIGN SMITH WM$IDENT
```

This command only needs to be executed once per login session but needs to be executed at least once to activate the new COMPOSER+ logical terminal for your session (most conveniently via the LOGIN.COM file).

To use the alternate character set with a VT220AC type terminal you will need to execute the following DCL command.

```
$ TYPE WM$DOWNLOAD:VTDEC.CFG
```

This command will download your terminal (VT200 & VT300-types) with the DECTEC characters that will be compatible with the LN03-PLUS and LN03R laser printers and will be displayed on your terminal when you select alternate characters.

After executing the above commands, COMPOSER+ can be activated by using the following command: (use the return key to execute DCL commands)

```
$ WM (or WMC)
```

Your screen should look something like the following:

WordMARC Composer+

V6.1.3

MAIN MENU

EDIT - - document or list
CREATE new document or list
PRINT or preview
AUXILRY manage print queue; more
FILES management
LANGUAGE change
QUIT WordMARC

Press 1st letter of choice, EXECUTE
[c] 1988 MARC Software International, Inc.

If you get an error then you have failed to correctly define WM\$IDENT or your username has not been added to the COMPOSER+ users file (see the system manager).

When you login to the system the SYLOGIN.COM file is executed. Several other logical assignments and symbols (e.g. WM\$EXE, WM\$FORMAT, WMC, etc.) are defined to initialize users' login session for COMPOSER+. Most users will not be concerned with the logical assignments made unless they choose to develop their own specially-prepared menus and formats to use with COMPOSER+.

The following file is executed by SYSTARTUP.COM when the MicroVAX II is booted.

```
$ ! This is the WMASSIGN.COM file
$ !
$ ! WordMARC logical name assignments
$ !
$ DEFINE/SYSTEM WM$MISC          $DISK3:[NEWWM.MISC]
$ !
$ DEFINE/SYSTEM WM$RUN           $DISK3:[NEWWM.RUN]
$ DEFINE/SYSTEM WM$RUNDICT       $DISK3:[NEWWM.RUNDICT]
$ DEFINE/SYSTEM WM$DOWNLOAD     $DISK3:[NEWWM.DOWNLOAD]
$ DEFINE/SYSTEM WM$DEVICE       $DISK3:[NEWWM.DEVICE]
$ DEFINE/SYSTEM WM$EXE          $DISK3:[NEWWM.EXE]
$ DEFINE/SYSTEM WM$FONT         $DISK3:[NEWWM.FONT.]
$ DEFINE/SYSTEM WM$FORMAT       $DISK3:[NEWWM.FORMAT]
$ DEFINE/SYSTEM WM$HELP        $DISK3:[NEWWM.HELP]
$ DEFINE/SYSTEM WM$TERM        $DISK3:[NEWWM.TERM]
$ DEFINE/SYSTEM WM$TYPE        $DISK3:[NEWWM.TYPE.]
$ DEFINE/SYSTEM WM$USER        SYS$LOGIN:
$ DEFINE/SYSTEM WM$DEFAULT      .WM
$ DEFINE/SYSTEM WM$LIST        .LIS
$ DEFINE/SYSTEM WM$TERMINAL     VT220AC
$ DEFINE/SYSTEM WM$PRINTER      POST
$ !
$ ! Note that the following symbols will not be used
$ ! when WM is running they are used with WM$TOOLS
$ ! command files
$ !
$ DEFINE/SYSTEM WM$TOP          $DISK3:[NEWWM]
$ DEFINE/SYSTEM WM$PRINT       $DISK3:[NEWWM.PRINT]
$ DEFINE/SYSTEM WM$MESSG       $DISK3:[NEWWM.MSG]
$ DEFINE/SYSTEM WM$FORM        $DISK3:[NEWWM.FORM]
$ DEFINE/SYSTEM WM$MENU        $DISK3:[NEWWM.MENU]
$ DEFINE/SYSTEM WM$HYPHEN      $DISK3:[NEWWM.HYPHEN]
$ DEFINE/SYSTEM WM$BINARY      $DISK3:[NEWWM.BINARY]
$ DEFINE/SYSTEM WM$TOOLS       $DISK3:[NEWWM.TOOLS]
$ DEFINE/SYSTEM WM$TRAINING    $DISK3:[NEWWM.TRAINING]
$ DEFINE/SYSTEM WM$MAIN        $DISK3:[NEWWM.MAIN]
$ DEFINE/SYSTEM WM$LOCAL       $DISK3:[NEWWM.LOCAL]
$ ! End of WMASSIGN file
```

Experienced users can, if they desire, still assign and use their own logical assignments and their own control files for COMPOSER+ applications. New users are recommended to stick to the "Learner's Guide" and the default logical configuration listed above.

The following file is executed when you login to the MicroVAX system.

```
$ !
$ ! SYMBOLS ASSIGNMENTS FOR SHARED WORDMARC
$ !
$ WM*C      :== $WM$EXE:WM.EXE
$ WMTRAN    :== $WM$EXE:WMTRAN.EXE
$ WMTRM     :== $WM$EXE:WMTRM.EXE
$ WMDICT    :== $WM$EXE:WMDICT.EXE
$ WMFORM    :== $WM$EXE:WMFORM.EXE
$ WMKEYS    :== $WM$EXE:WMKEYS.EXE
$ WMMENU    :== $WM$EXE:WMMENU.EXE
$ WMMSG     :== $WM$EXE:WMMSG.EXE
$ WMHYP     :== $WM$EXE:WMHYP.EXE
$ WMPOST    :== $WM$EXE:WMPOST.EXE
$ WMTSLT    :== $WM$EXE:WMTSLT.EXE
$ WMUTIL    :== @WM$TOOLS:WMUTIL.COM
$ WM$MAIL   :== MAIL
$ !
$ LM        :== $WM$EXE:LM.EXE
```

This file is executed by SYLOGIN.COM before your LOGIN.COM file is executed. The different symbols are defined to provide users short commands and access to the different COMPOSER+ features. Most users will only use the WM or WMC symbols to execute the WORDMARC COMPOSER+ program for scientific word processing applications. Refer to the user documentation and the VMS HELP facility for additional information on using COMPOSER+.

The MicroVAX HELP facility has been updated to provide online help for users; type the following command to get online help for using the new COMPOSER+ software.

\$ HELP WM

Official documents describe the features of WORDMARC COMPOSER+:

1. The "Learner's Guide" is what most users will need first.
2. The "Reference Guide" contains detailed information.
3. The "Quick Reference Guide" is small and useful after users become knowledgeable of COMPOSER+ features.
4. The "Technical Reference Manual" is mostly for installation but contains some information regarding the guts of COMPOSER+.

The following users have been added to the COMPOSER+ terminal definition file located in WM\$MISC:TCONFIG.CFG. This additional information is provided to only help users get oriented and to use COMPOSER+ effectively.

BAKER VT220AC SPEED 10
BERTRAND VT220AC SPEED 10
BRAYTON VT220AC SPEED 10
BRYSON VT220AC SPEED 10
COTTER VARIABLE SPEED 10
DAWBARN VT220AC SPEED 10
EARL LOGICAL SPEED 10
ELROD VARIABLE SPEED 10
FEDER VT220AC SPEED 10
GOETHERT VARIABLE SPEED 10
HERRON VT220AC SPEED 10
HOLT VARIABLE SPEED 10
JIM VT220AC SPEED 10
JOHNSON VT220AC SPEED 10
JONES VT220AC SPEED 10
LINDEMAN VT220AC SPEED 10
LOWRY VT220AC SPEED 10
MEAD VT220AC SPEED 10
MOORE VT220AC SPEED 10
NGUYEN VT220AC SPEED 10
NICHOLSON VT220AC SPEED 10
PRICE VT220AC SPEED 10
RICHARDS VT220AC SPEED 9600
SEIBER VT220AC SPEED 10
SELMAN VARIABLE SPEED 10
SIMPSON VT220AC SPEED 10
SISCO VT220AC SPEED 10
SMITH VT220AC SPEED 10
STEELE VT220AC SPEED 10
STEELY VARIABLE SPEED 10
THAD VR220AC SPEED 10
WARMBROD VT220AC SPEED 10
WHITE VARIABLE SPEED 10
WILLIAMS VT220AC SPEED 10
WOOD VT220AC SPEED 10
YOUNG VT220AC SPEED 10

The first entry corresponds to your 'username' that is associated with the assignment you make (ASSIGN username WM\$IDENT) before using COMPOSER+.

The second entry corresponds to one of the terminal definitions that are available for use. If yours is VARIABLE then you will be prompted for the type of terminal definition you want to use, otherwise your terminal definition will default to the second entry, e.g. VT220AC. If you use different terminal types (PCs, VT200's, etc.) then you will need to have a VARIABLE terminal definition.

Experienced users can create their own menus, formats, etc. You then have to reassign the logicals to correctly point to the format files, menus, etc. This allows users to tailor their menus and formats for special applications. New users can use the default logical assignments and do not and should not contend with the advanced features provided. The default configuration will be satisfactory for most user applications.

Generating your own menus is a personal preference. You can use the menus supplied by COMPOSER+ by just using the procedure described on page one. Creating your own menus and formats is discussed in the documentation.

As examples of the use of COMPOSER+, the following pages illustrate some of the locally added features using the LN03-PLUS and the LN03R laser printers' attributes. This USER INFORMATION document was created using a typeset format with the CENT12 postscript font selection. It was printed on printer P1, a Digital LN03R ScriptPrinter postscript printer.

The old print definitions have been retained for compatibility; users will want to become familiar with the new postscript printer and the available fonts. The local COMPOSER+ font selections have been reduced to those compatible with the LN03R postscript printer.

The following printer definitions are provided.

```
# 1, PORTRAIT, ELITE, 12CPI, DECTEC-ALTERNATE, LN03 SYS$PRINT
# 2, PORTRAIT, COUR., 10CPI, DECTEC-MULTINAT., LN03 SYS$PRINT
# 3, LANDSCAPE, 14 POINT, 8 PITCH, GOTHIC FONT, LN03 SYS$PRINT
# 4, PORTRAIT, 10 POINT C.G. TIMES, LN03 SYS$PRINT (C.G. TIMES ROM REQUIRED)
# 5, PORTRAIT, 12 POINT C.G. TIMES, LN03 SYS$PRINT (C.G. TIMES ROM REQUIRED)
# 6, LANDSCAPE, 6.7 POINT, 13.6CPI, 64 VERTICAL, 108 COLUMNS, LN03 SYS$PRINT
# 7, LANDSCAPE, 18 POINT TRIUM., LN03 SYS$PRINT (DOWNLOADED ON RAM CART.)
# 8, PORTRAIT, C.G. TIMES, 10 POINT + OTHER TYPEFACES, LN03 SYS$PRINT
# 9, LANDSCAPE, ELITE, 12CPI, LN03 SYS$PRINT (MUST HAVE RAM TO ROTATE FONT!)
#10, PORTRAIT, ELITE + TYPEFACE, USES 12cpi for COUR6, LN03 SYS$PRINT
#11, LANDSCAPE, 6.7 POINT, 13.6 CPI, 66 VER. LINES, 145 COL., LN03 SYS$PRINT
#12, PORTRAIT, 6.7 POINT, 13.6 CPI, 87 VER. LINES, 132 COL., LN03 SYS$PRINT
#13, PORTRAIT, ELITE, 12CPI, DECTEC-ALTERNATE, LN03 SYS$LN03
#14, LANDSCAPE, ELITE + TYPEFACE, USES 12cpi for COUR6, LN03 SYS$PRINT

#P1, LANDSCAPE OR PORTRAIT, MANY FONTS, LN03R POSTSCRIPT PRINTER, SYS$LN03R
#P2, LANDSCAPE OR PORTRAIT, MANY FONTS, LN03R POSTSCRIPT PRINTER, SYS$LN03R2
#EKPP, LANDSCAPE OR PORTRAIT, MANY FONTS, LN03R POSTSCRIPT PRINTER, EKPP
```

Printer definitions 1-14 have been retained for the LN03-PLUS laser printer; P1, P2, and EKPP have been set up to support the typestyle printer document formats for printing on a postscript laser printer.

It should be noted that the CREATE option of the main menu selects preconfigured formats (files) that will contain a printer option selection that corresponds to one of the above printer numbers by default. You can change the printer option number but the results of your printed output may not correspond to the preconfigured format you chose to create your document with. You can, none the less, mix and match to your own satisfaction. It is recommended that you experiment with the output to get a feel for the type of format and characters you desire.

Use the default typestyles provided if you create a typestyle-type document; select the TYPE COMPOSER+ function to activate the font selection!! The following postscript fonts are available on the LN03R ScriptPrinter and available for selection from within a COMPOSER+ typestyle document format.

<u>LN03R PRINTER NAME</u>	<u>COMPOSER+ NAME</u>
AvanteGarde-Book	avgb**n
AvanteGarde-BookOblique	avgb**i
AvanteGarde-Demi	avgb**b
AvanteGarde-DemiOblique	avgb**q
Courier	cour**n
Courier-Bold	cour**b
Courier-Oblique	cour**i
Courier-BoldOblique	cour**q
DEC-Tech	dtch**n
Helvetica	helv**n
Helvetica-Bold	helv**b
Helvetica-Oblique	helv**i
Helvetica-BoldOblique	helv**q
LubalinGraph-Book	lubl**n
LubalinGraph-BookOblique	lubl**i
LubalinGraph-Demi	lubl**b
LubalinGraph-DemiOblique	lubl**q
NewCentrySchlbk-Roman	cent**n
NewCentrySchlbk-Bold	cent**b
NewCentrySchlbk-Italic	cent**i
NewCentrySchlbk-BoldItalic	cent**q
Souvenir-Demi	souv**b
Souvenir-DemiItalic	souv**q
Souvenir-Light	souv**n
Souvenir-LightItalic	souv**i
Symbol	symb**n
Times-Roman	trom**n
Times-Bol	trom**b
Times-Italic	trom**i
Times-BoldItalic	trom**q

Where ** is chosen to be 06, 08, 10, 12, 16, 18, 24, 36, or 72 (font size in points).

Right justification is now supported with the postscript proportional character fonts; you can now use proportional characters with right justification, it results in a clean justified edge. Proportional character fonts, however, still do not

work well with mathematical equations!!! The monospaced (Courier) font should be used when complex equations are generated.

The following keyboard template was created using the cour10n and dtch10n COMPOSER+ font selection. The DECTEC characters are supported on the LMPCC (PCs) and VT220AC terminals and the LN03-PLUS and LN03R printers.

DECTECHNICAL CHARACTER TEMPLATE	
LOWER CASE	UPPER CASE
`1234567890-=_	~!@#\$%^&*()_+
~^_√-~> }≠	↓ :-)Λ √{
qwertyuiop[]	QWERTYUIOP{}
ψωερτϑ ιδπς	ΨΩV?QTC=≡Π←→
asdfghjkl;'\"	ASDFGHJKL:"
ασδφγηθκλ [ο	≈ΣΔΦΓ~Θ×Λ ρ↑
<zxcvbnm,./	>ZXCVBNM,.?
zξχfβνμ U{	zCΞ+√∞≈ U}

Capital Greek letters available (some are not alternate characters)

A B Γ Δ E Z H Θ I K Λ M N Ξ O Π P Σ T Y Φ X Ψ Ω
A B G D E Z H J I K L M N X O P P S T Y F X Q W

Lower case Greek letters available (o is a regular character)

α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ σ τ υ φ χ ψ ω
a b g d e z h j i k l m n x o p r s t y f c q w

Additional math and logic symbols

= ≈ ~ ≥ ≤ ≠ ∞ ← ↑ ↓ → ^ ∨ ⊃ ∪ ∩ ⊂ × + ⇒ ⇐ ∴
O I H > < = A { | ~ } ^ _ [] \ Z K C N M @

Other math symbols

DEL V	Infinity ∞	Integral ∫	Partial ∂	Script-f f	Square root √
E	B	?	o	v	v

Composite math symbols

Left Brace { [	Right Brace }]
, & +	. & -
Left Bracket [[	Right Bracket]]
(& '	* &)
left curly Brace { { {	Right curly Brace } } }
, / +	. 0 -

Integral ∫ | [

. & +

Summation ∑ > ~ ^ \ / -

6 7 5 1 2 3 4 #

As an example of the use of a local glossary the following template can be used for reference and can be copied from \$DISK3:[GENERAL]WAGLOS.CFG

LOCAL GLOSSARY ITEMS FOR MATH SYMBOLS

LEFT AND RIGHT PARENTHESIS:

LP1	LP2	LP3	LP4	LP5	LP6	RP6	RP5	RP4	RP3	RP2	RP1
↓	↓	↓	↓					↓	↓	↓	↓
(	(	(	(	(	(	)	)	)	)	)	)

LEFT AND RIGHT BRACKETS:

LBK1	LBK2	LBK3	LBK4	LBK5	LBK6	RBK6	RBK5	RBK4	RBK3	RBK2	RBK1
↓	↓	↓	↓					↓	↓	↓	↓
[	[	[	[	[	[	]	]	]	]	]	]

LEFT AND RIGHT CURLY BRACES:

LCB1	LCB2	LCB3	LCB4	LCB5	LCB6	RCB6	RCB5	RCB4	RCB3	RCB2	RCB1
↓	↓	↓	↓					↓	↓	↓	↓
{	{	{	{	{	{	}	}	}	}	}	}

INTEGRALS:

I1	I2	I3	I4	I5	I6	CI1	CI2	CI3	CI4	CI5	CI6
↓	↓	↓	↓			↓	↓	↓	↓	↓	↓
∫	∫	∫	∫	∫	∫	∫	∫	∫	∫	∫	∫

SUMMATION:

S1	S2	S3	S4	S5	S6
↓	↓	↓	↓	↓	
Σ	Σ	Σ	Σ	Σ	Σ

Being a scientific word processor, COMPOSER+ has been specially tuned to support mathematical copy. As an example, the complex Wierstrass function

$$w_0(t) = \left(1-w^2\right)^{-1/2} \sum_{n=0}^{\infty} w^n \exp(2\pi i b^n t),$$

where b is a real number > 1 , and

$$w = \begin{cases} b^H & , 0 < H < 1 \\ b^{D-2} & , 1 < D < 2 \end{cases}$$

is a continuous function that has no derivative at any point within it's domain and conceptually transcends one's mental visual capacity.

The divergence theorem states that for any well-behaved vector field $A(x)$ defined within a volume V surrounded by the closed surface S the relation

$$\oint_S \vec{A} \cdot \vec{n} \, da = \int_V \nabla \cdot \vec{A} \, d^3x$$

holds between the volume integral of the divergence of A and the surface integral of the outwardly directed normal component of A . In tensor notation we would write

$$\oint_S A_i n_i \, da = \int_V \partial_i A_i \, d^3x$$

Other font sizes are available also, e.g., the identity projection operator is defined as follows

$$I \equiv \sum |i \rangle \langle i|$$

$$|\Psi\rangle = I|\Psi\rangle = (\sum |i\rangle \langle i|)|\Psi\rangle = \sum |i\rangle \langle i|\Psi\rangle$$

Other examples are relegated to the reader/user.

Slatec 3.2

User Information

SLATEC Common Math Library

SLATEC is a large collection of **FORTTRAN** mathematical subroutines brought together in a joint effort by the Air Force Weapons Laboratory, Lawrence Livermore National Laboratory, Los Alamos National Laboratory, Magnetic Fusion Energy Computing Center, National Bureau of Standards, Sandia National Laboratories (Albuquerque and Livermore), and the Martin Marietta Energy Systems Incorporated at Oak Ridge National Laboratory.

SLATEC is characterized by portability, good numerical technology, good documentation, robustness, and quality assurance. The Library can be divided into the following subsections following the lines of the NBS classification system: Elementary and Special Functions, Elementary Vector Operations, Solutions of Systems of Linear Equations, Eigenanalysis, QR Decomposition, Singular Value Decomposition, Interpolation, Solution of Nonlinear Equations, Optimization, Quadrature, Ordinary Differential Equations, Partial Differential Equations, Fast Fourier Transforms, Approximations, Psuedo-random Number Generation, Sorting, Machine Constants, and Diagnostics and Error Handling.

The **SLATEC** subroutines have been compiled and stored in the library file **DISK3:[SLATEC]SLATEC.OLB**. The library file is available for users to link with their programs. For example you link your source object file with the **SLATEC** library using the **LINK** command:

\$ LINK your_object_file,\$DISK3:[SLATEC]SLATEC.OLB/L

The original source code is also available to users along with a program **\$DISK3:[SLATEC]SLAT.EXE** to extract the desired subroutine(s) and all unresolved externals. The source code is sometimes useful to include in your program to increase its portability and completeness. A modified **HELP** program is stored as **\$DISK3:[SLATEC]SLATHELP.EXE**.

Both programs can be used and are accessible through the command file **\$DISK3:[SLATEC]SLATEC.COM** generated for general use. Users simply type **SLATEC** to execute this command file after logging into the system, e.g.,

\$ SLATEC

Users needing specific information about the **SLATEC** help file or subroutines should contact Sid Steely (ext. 7730) if you have questions.

The **SLATEC 3.2** Library is summarized on the following pages. As part of the CN45VW project effort, the **SLATEC 3.2** files were obtained from the Air Force Weapons Laboratory for use at AEDC on the Space Systems Ethernet LAVC Network computer systems, ELA-1077.

Contents of The SLATEC Library

Category A	Arithmetic, Error Analysis
Category C	Elementary and Special Functions
Category D1a	Elementary Vector Operations
Category D2	Solution of Systems of Linear Equations
Category D4	Eigenvalues and Eigenvectors
Category D5	QR Decomposition
Category D6	Singular Value Decomposition
Category D9	Overdetermined or Underdetermined Systems, Pseudo-Inverses
Category E	Interpolation
Category F	Zeros of Functions/Solution of Nonlinear Equations
Category G	Optimization
Category H2	Quadrature (Numerical Evaluation of Definite Integrals)
Category I1	Ordinary Differential Equations
Category I2	Partial Differential Equations
Category J1	Fast Fourier Transforms
Category K	Approximation
Category L6	Pseudo-Random Number Generation
Category N6	Sorting
Category R1	Machine-dependent Constants
Category R3	Error Handling
Category Z	Other

14	15A	ARITHMETIC, ERROR ANALYSIS
16	25C	ELEMENTARY AND SPECIAL FUNCTIONS
26	47D	LINEAR ALGEBRA
48	52E	INTERPOLATION
53	55F	SOLUTION OF NONLINEAR EQUATIONS
56	62H	DIFFERENTIATION AND INTEGRATION
63	71I	DIFFERENTIAL AND INTEGRAL EQUATIONS
72	72J	INTEGRAL TRANSFORMS
73	77K	APPROXIMATION
78	78L	STATISTICS AND PROBABILITY
79	79N	DATA HANDLING
80	81R	SERVICE ROUTINES
13	13Z	OTHERS
	A4	Complex
	A6	Change of representation
	C1	Integer-valued functions (e.g., floor, ceiling, factorial, binomial coefficient)
	C2	Powers, roots, and reciprocals
	C3	Polynomials
	C4	Elementary transcendental functions
	C5	Exponential and logarithmic integrals
	C7	Gamma
	C8	Error functions
	C10	Bessel functions
	C11	Confluent hypergeometric functions
	D1	Elementary vector and matrix operations
	D1A	Elementary vector operations
	D1B	Elementary matrix operations
	D2	Solution of systems of linear equations (including inversion and LU and related decompositions)
	D2A	Real non-symmetric matrices
	D2B	Real symmetric matrices
	D2C	Complex non-Hermitian matrices
	D2D	Complex Hermitian matrices
	D3	Determinants
	D3A	Real non-symmetric matrices
	D3B	Real symmetric matrices
	D3C	Complex non-Hermitian matrices
	D3D	Complex Hermitian matrices
	D4	Eigenvalues and eigenvectors
	D4A	Ordinary eigenvalue problems ($Ax = \lambda x$)
	D4B	Generalized eigenvalue problems (e.g., $Ax = \lambda Bx$)
	D5	QR decomposition, Gram-Schmidt orthogonalization
	D6	Singular value decomposition
	D7	Update matrix decomposition
	D9	Overdetermined or underdetermined systems of equations, singular systems, pseudo-inverses (also see D5, D6, and K1A)
	E1	Univariate data (curve fitting)

E1A	Polynomial splines (piecewise polynomials)
E1B	Polynomials
E3	Service routines (e.g., grid generation, evaluation of fitted functions)
F1	Single equation
F2	System of equations
F3	Service routines (e.g., check user-supplied derivatives)
H2	Quadrature (numerical evaluation of definite integrals)
H2A	One dimensional integrals
H2A1	Finite-interval (general integrand)
H2A2	Finite-interval (specific or special type integrand including weight functions, oscillating and singular integrands, principal value integrals, splines, etc.)
H3A3	Semi-infinite interval (including $\exp(-x)$ weight function)
I1	Ordinary differential equations
I1A	Initial value problems
I1A1	General, non-stiff or mildly stiff
I1A2	Stiff and mixed algebraic-differential equations
I1B	Multi-point boundary value problems
I2	Partial differential equations
I2B	Elliptic boundary value problems
I2B1	Linear
I2B4	Service routines
J1	Fast Fourier transforms
K1	Least squares approximation
K1A	Linear least squares (also see D5, D6, and D9)
K1B	Nonlinear least squares
K6	Service routines (e.g., mesh generation, evaluation of fitted functions)
L6	Pseudo-random number generation
N6	Sorting
R1	Machine-dependent constants
R3	Error handling

Category A Arithmetic, Error Analysis

single double complx integr

R9PAK D9PAK - Pack a base 2 exponent into a floating point number.

R9UPAK D9UPAK - Unpack a floating point number X so that $X=Y*2^{**N}$.

Category C Elementary and Special Functions

single double complx integr

- - CACOS Compute the complex arc Cosine.
- - CARG Compute the argument of a complex number.
- - CASIN Compute the complex arc Sine.
- - CATAN Compute the complex arc Tangent.
- - CATAN2 Compute the complex arc Tangent in the proper quadrant.
- - CCOSH Compute the complex hyperbolic Cosine.
- - CLOG10 Compute the principal value of the complex base 10 log.
- - CSINH Compute the complex hyperbolic Sine.
- - CTAN Compute the complex Tangent.
- - CTANH Compute the complex hyperbolic Tangent.

ACOSH DACOSH CACOSH Compute the arc hyperbolic Cosine.

AI DAI - Compute the Airy function.

AIE DAIE - Compute the exponentially scaled Airy function.

ALBETA DLBETA CLBETA Compute the natural log of the complete Beta function.

ALGAMS DLGAMS - Compute the log of the absolute value of the Gamma function including the sign.

ALI DLI - Compute the logarithmic integral.

ALNGAM DLNGAM CLNGAM Compute the log of the absolute value of the Gamma

function.

ALNREL DLNREL CLNREL Compute $\ln(1+X)$ accurate in the sense of relative error.

ASINH DASINH CASINH Compute the arc hyperbolic Sine.

ATANH DATANH CATANH Compute the arc hyperbolic Tangent.

BESI DBESI - Compute an N member sequence of Bessel functions $I_{\text{sub}}(\text{ALPHA}+K-1)$ at X , $K=1, \dots, N$ or scaled Bessel functions $\text{EXP}(-X) * I_{\text{sub}}(\text{ALPHA}+K-1)$ at X , $K=1, \dots, N$ for non-negative ALPHA and X .

BESIO DBESIO - Compute the hyperbolic Bessel function of the first kind of order zero.

BESIOE DBESIOE - Compute the exponentially scaled hyperbolic Bessel function of the first kind of order zero.

BESI1 DBESI1 - Compute the hyperbolic Bessel function of first kind of order one.

BESI1E DBESI1E - Compute the exponentially scaled hyperbolic Bessel function of the first kind of order one.

BESJ DBESJ - Compute an N member sequence of J Bessel functions $J_{\text{sub}}(\text{ALPHA}+K-1)$ at X , $K=1, \dots, N$ for non-negative ALPHA and X .

BESJ0 DBESJ0 - Compute the Bessel function of the first kind of order zero.

BESJ1 DBESJ1 - Compute the Bessel function of the first kind of order one.

BESK DBESK - Implements forward recursion on the three term recursion relation for a sequence of non-negative order Bessel functions $K_{\text{sub}}(\text{FNU}+I-1)$, or scaled Bessel functions $\text{EXP}(X) * K_{\text{sub}}(\text{FNU}+I-1)$ at X , $I=1, \dots, N$ for $X > 0.0\text{E}0$ and non-negative orders FNU.

BESK0 DBESK0 - Compute the hyperbolic Bessel function of the third kind of order zero.

BESK0E DBSK0E - Compute the exponentially scaled hyperbolic Bessel function of the third kind of order zero.

BESK1 DBESK1 - Compute the hyperbolic Bessel function of the third kind of order one.

BESK1E	DBSK1E	-	Compute the exponentially scaled hyperbolic Bessel function of the third kind of order one.
BESKES	DBSKES	-	Compute a sequence of exponentially scaled modified Bessel functions of the third kind of fractional order.
BESKS	DBESKS	-	Compute a sequence of modified Bessel functions of the third kind of fractional order.
BESY	DBESY	-	Implements forward recursion on the three term recursion relation for a sequence of non-negative order Bessel functions $Y_{\text{sub}(FNU+I-1)}$ at X , $I=1,N$ for real X $.gt.0$ and for non-negative orders FNU .
BESY0	DBESY0	-	Compute the Bessel function of the second kind of order zero.
BESY1	DBESY1	-	Compute the Bessel function of the second kind of order one.
BETA	DBETA	CBETA	Compute the complete Beta function.
BETAI	DBETAI	-	Compute the incomplete Beta function.
BI	DBI	-	Compute the Bairy function.
BIE	DBIE	-	Compute the exponentially scaled Bairy function.
BINOM	DBINOM	-	Compute the binomial coefficients.
BSKIN	DBSKIN	-	Compute repeated integrals of the K-zero Bessel function.
CBRT	DCBRT	CCBRT	Compute the cube root of the argument.
CHU	DCHU	-	Compute the logarithmic confluent hypergeometric function.
COSDG	DCOSDG	-	Compute the Cosine of an argument in degrees.
COT	DCOT	CCOT	Compute the complex Cotangent.
CSEVL	DCSEVL	-	Evaluate the N-term Chebyshev series.
DAWS	DDAWS	-	Compute Dawson's function.
E1	DE1	-	Compute the exponential integral $E1(X)$.
EI	DEI	-	Compute the exponential integral $EI(X)$.

ERF	DERF	-	Compute the error function.
ERFC	DERFC	-	Compute the complementary error function.
EXINT	DEXINT	-	Compute M member sequences of exponential integrals $E(N+K, X)$, $K=0, 1, \dots, M-1$ for $N \geq 1$ and $X \geq 0$.
EXPREL	DEXPRL	CEXPRL	Evaluate $EXPREL(X) = (EXP(X)-1)/X$.
FAC	DFAC	-	Compute the factorial of N.
GAMI	DGAMI	-	Compute the incomplete Gamma function.
GAMIC	DGAMIC	-	Compute the complementary incomplete Gamma function.
GAMIT	DGAMIT	-	Compute Tricomi's form of the incomplete Gamma function.
GAMLIM	DGMLM	-	Compute the minimum and maximum bounds for GAMMA.
GAMMA	DGAMMA	CGAMMA	Compute the complete Gamma function.
GAMR	DGAMR	CGAMR	Compute the reciprocal Gamma function.
INITS	INITDS	-	Initialize to determine the number of terms to carry in an orthogonal series to meet a specified error.
POCH	DPOCH	-	Compute a generalization of Pochhammer's symbol.
POCH1	DPOCH1	-	Compute a generalization of Pochhammer's symbol starting from first order.
PSI	DPSI	CPSI	Compute the Psi (or Digamma) function.
PSIFN	DPSIFN	-	Compute derivatives of the Psi function.
-	-	COLGMC	Evaulates $(z+0.5)*CLOG((Z+1.)/Z) - 1.0$ with relative accuracy.
R9AIMP	D9AIMP	-	Evaluates the Airy modulus and phase for $X \leq -1.0$.
R9ATN1	D9ATN1	-	Evaluate $ATAN(X)$ from first order relative accuracy so that $ATAN(X) = X + X**3*R9ATN1(X)$.
R9CHU	D9CHU	-	Evaulates for large Z, $Z**A * U(A,B,C)$, where U is the logarithmic confluent hypergeometric function.
-	D9B0MP	-	Evaluates the modulus and phase for J0 and K0 Bessel functions.

- D9B1MP - Evaluates the modulus and phase for J1 and K1 Bessel functions.

R9GMIC D9GMIC - Calculates the complementary incomplete Gamma function for A near a negative integer and X small.

R9KNUS D9KNUS - Computes Bessel functions $\text{EXP}(X)^{K-\text{SUB}-XNU(X)}$ and $\text{EXP}(X)^{K-\text{SUB}-XNU+1(X)}$ for $0.0 \leq XNU < 1.0$.

R9GMIT D9GMIT - Computes Tricomi's incomplete Gamma function for small X.

R9LGIC D9LGIC - Computes the log complementary incomplete Gamma function for large X and $A \leq X$.

R9LGIT D9LGIT - Computes the log of Tricomi's incomplete Gamma function with Perron's continued fraction for large X and $A \geq X$.

R9LGMC D9LGMC C9LGMC Compute the log Gamma correction factor so that $\text{ALOG}(\text{GAMMA}(X)) = \text{ALOG}(\text{SQRT}(2*\text{PI})) + (X-.5)*\text{ALOG}(X) - X + \text{R9LGMC}(X)$.

R9LN2R D9LN2R C9LN2R Evaluate $\text{ALOG}(1+X)$ from second order relative accuracy so that $\text{ALOG}(1+X) = X - X^2/2 + X^3*\text{R9LN2R}(X)$.

RC DRC - Compute the Elliptic integral defined as the integral from zero to infinity of $.5*dt/((t+Y)*\text{sqrt}(t+X))$.

RD DRD - Compute the incomplete or complete Elliptic integral of the second kind.

RF DRF - Compute the incomplete or complete Elliptic integral of the first kind.

RJ DRJ - Compute the incomplete or complete Elliptic integral of the third kind.

SINDG DSINDG - Compute the Sine of an argument in degrees.

SPENC DSPENC - Compute a form of Spence's integral due to K. Mitchell.

Category D1a Elementary Vector Operations

single double complx integr

ISAMAX IDAMAX ICAMAX Find the largest component of a vector.

SASUM	DASUM	SCASUM	Sum of the magnitudes of vector components.
SAXPY	DAXPY	CAXPY	Computation (for scalar a) of $y = a*x + y$.
SCOPY	DCOPY	CCOPY	Vector copy $y = x$.
SCOPYM	DCOPYM	-	Vector copy (with sign change) $y = -x$.
SDOT	DDOT	CDOTU	Vector inner (dot) product.
-	-	CDOTC	Dot product of complex vectors using complex conjugate of the first vector.
SDSDOT	-	CDCDOT	Vector dot product with double precision accumulation.
-	DSDOT	DCDOT	Double precision dot product of single precision vectors
SNRM2	DNRM2	SCNRM2	Euclidean length (L2 norm) of a vector.
SROT	DROT	CSROT	Apply a plane Given's rotation.
SROTG	DROTG	CROTG	Construct a plane Given's rotation.
SROTM	DROTM	-	Apply a modified Given's transformation.
SROTMG	DROTMG	-	Construct a modified Given's transformation.
SSCAL	DSCAL	CSCAL	Vector scale $x = a*x$.
-	-	CSSCAL	Scale a complex vector.
SSWAP	DSWAP	CSWAP	ISWAP Vector interchange.
-	DQDOTA	-	Inner product with extended precision accumulation and result.
-	DQDOTI	-	Inner product with extended precision accumulation and result.

Category D2 Solution of Systems of Linear Equations

single double complx integr Also see LINPACK routines.

SGEFS	DGEFS	CGEFS	Solve a general real (double, complex) NxN system of linear equations.
-------	-------	-------	--

SGEIR	-	CGEIR	Solve a general real (complex) $N \times N$ system of linear equations. Iterative refinement is used to obtain an error estimate.
SNBCO	DNBCO	CNBCO	Factors a BAND matrix by Gaussian elimination and estimates the condition of the matrix.
SNBFA	DNBFA	CNBFA	Factors a BAND matrix by Gaussian elimination.
SNBFS	DNBFS	CNBFS	Solve a general nonsymmetric banded real (double,complex) $N \times N$ system of linear equations.
SNBIR	-	CNBIR	Solve a general nonsymmetric banded real (complex) $N \times N$ system of linear equations. Iterative refinement is used to obtain an error estimate.
SNBSL	DNBSL	CNBSL	Solves a Banded system using factors created by SNBCO or SNBFA, or by DNBCO or DNBFA, or by CNBCO or CNBFA.
SPOFS	DPOFS	CPOFS	Solve a positive definite symmetric real (double,complex) $N \times N$ system of linear equations.
SPOIR	-	CPOIR	Solve a positive definite real symmetric (complex Hermitian) $N \times N$ system of linear equations. Iterative refinement is used to obtain an error estimate.

Category D3 Determinants

single double complx integr

SNBDI	DNBDI	CNBDI	Computes the determinant of a BAND matrix using factors created by SNBCO or SNBFA, or by DNBCO or DNBFA, or by CNBCO or CNBFA.
-------	-------	-------	--

Category D4 Eigenvalues and Eigenvectors (See EISPACK)

single double complx integr Also see Eispack routines.

EISDOC Documentation for the EISPACK eigenvalue package.

SGEEV	-	CGEEV	Compute the eigenvalues and, optionally, the eigenvectors of a real (complex) general matrix.
-------	---	-------	---

SSIEV	-	CHIEV	Compute the eigenvalues and, optionally, the eigenvectors of a real symmetric (complex Hermitian) matrix.
-------	---	-------	---

SSPEV - - Compute the eigenvalues and, optionally, the eigenvectors of a real symmetric matrix stored in packed form.

Category D5 QR Decomposition

single double complx integr

SQRDC DQRDC CQRDC Use Householder transformations to compute the QR factorization of an NxP real (double,complex) matrix. Column pivoting is optional.

SQRSL DQRSL CQRSL Apply the output of *QRDC to compute coordinate transformations, projections, and least squares solutions.

Category D6 Singular Value Decomposition

single double complx integr

SSVDC DSVDC CSVDC Perform the singular value decomposition of a real (double,complex) NxP matrix.

Category D9 Overdetermined or Underdetermined Systems, Pseudo-inverses

single double complx integr

BNDACC DBNDAC - Together with BND SOL, solve the least squares problem $Ax=b$ for banded matrices A using sequential accumulation of the rows of the data matrix A.

BND SOL DBND SL - Together with BNDACC, solve the least squares problem $Ax=b$ for banded matrices A using sequential accumulation of the rows of the data matrix A.

HFTI DHFTI - Solve the linear least squares problem $AX=B$, where B is a matrix. If B is the MxM identity matrix, then X is the pseudo-inverse of A.

LLSIA DLLSIA - Solves LINEAR LEAST SQUARES problems by performing a QR factorization of the matrix A using Householder transformations. Emphasis is put on detecting possible rank deficiency.

- SGLSS DGLSS - Solve linear least squares problems by performing a QR factorization using Householder transformations. Emphasis is put on detecting possible rank deficiency.
- ULSIA DULSIA - Solves the UNDERDETERMINED LINEAR system of equations by performing an LQ factorization of the matrix A using Householder transformations. Emphasis is put on detecting possible rank deficiency.

Category E Interpolation

single double complx integr

- BFQAD DBFQAD - Compute the integral on (X1,X2) of a product of a function and the ID-th derivative of a K-th order B-spline (B-representation).
- BINT4 DBINT4 - Compute the B representation of a cubic spline which interpolates data (X(I),Y(I)), I=1,NDATA.
- BINTK DBINTK - Produce the B-spline coefficients, BCOEF, of the B-spline of order K with knots T(I), I=1,...,N+K, which takes on the value Y(I) at X(I), I=1,...,N.
- BSPPDR DBSPDR - Use the B-representation to construct a divided difference table preparatory to a (right) derivative calculation in BSPEV.
- BSPEV DBSPEV - Compute the value of the spline and its derivatives from the B-representation.
- BSPPP DBSPPP - Convert the B-representation to the piecewise polynomial (PP) form for use with PPVAL.
- BSPPVD DBSPVD - Compute the value and all derivatives of order less than NDERIV of all basis functions which do not vanish at X.
- BSPPVN DBSPVN - Compute the value of all (possibly) nonzero basis functions at X.
- BSQAD DBSQAD - Compute the integral on (X1,X2) of a K-th order B-spline using the B-representation.
- BVALU DBVALU - Evaluate the B-representation of a B-spline for the function value or any of its derivatives.
- CHFDV DCHFDV - Evaluate a cubic polynomial given in Hermite form and its first derivative at an array of points. While

designed for use by PCHFD, may be used directly as an evaluator in applications, such as graphing, where the interval is known in advance. If only function values are required, use CHFV instead.

- CHFV DCHFV - Evaluate a cubic polynomial given in Hermite form at an array of points. While designed for use by PCHFE, may be used directly as an evaluator in applications, such as graphing, where the interval is known in advance.

- INTRV DINTRV - Compute the largest integer ILEFT in the interval $[1, LXT]$ such that $XT(ILEFT) \leq X$ where $XT(*)$ is a subdivision of the X interval.

- PCHFD DPCHFD - Evaluate a piecewise cubic Hermite function and its first derivative at an array of points. May be used by itself for Hermite interpolation, or as an evaluator for PCHIM or PCHIC. If only function values are required, use PCHFE instead.

- PCHFE DPCHFE - Evaluate a piecewise cubic Hermite function at an array of points. May be used by itself for Hermite interpolation, or as an evaluator for PCHIM or PCHIC.

- PCHIA DPCHIA - Evaluate the definite integral of a piecewise cubic Hermite function over an arbitrary interval.

- PCHIC DPCHIC - Set derivatives needed to determine a piecewise monotone piecewise cubic Hermite interpolant to given data. User control is available over boundary conditions and/or the treatment of points where monotonicity switches direction.

- PCHID DPCHID - Evaluate the definite integral of a piecewise cubic Hermite function over an interval whose endpoints are data points.

- PCHIM DPCHIM - Set derivatives needed to determine a monotone piecewise cubic Hermite interpolant to given data. Boundary values are provided which are compatible with monotonicity. The interpolant will have an extremum at each point where monotonicity switches direction. Use PCHIC if control is desired over boundary or switch conditions.

- PCHDOC Documentation for the PCHIP routines. PCHIP is a Fortran package for piecewise cubic Hermite interpolation of data. It features software to produce a monotone and "visually pleasing" interpolant to monotone data.

PCHMC	DPCHMC	-	Check a cubic Hermite function for monotonicity.
PCHSP	DPCHSP	-	Produce a cubic spline interpolator in cubic Hermite form. Provided primarily for easy comparison of the spline with other piecewise cubic interpolants.
PFQAD	DPFQAD	-	Compute the integral of a product of a function and the ID-th derivative of a B-spline, (PP-representation).
POLINT	DPLINT	-	Produce the polynomial which interpolates a set of discrete data points.
POLYVL	DPOLVL	-	Calculate the value of the polynomial and its first NDER derivatives where the polynomial was produced by a previous call to POLINT.
PPQAD	DPPQAD	-	Compute the integral on (X1,X2) of a K-th order B-spline using the piecewise polynomial representation.
PPVAL	DPPVAL	-	Compute the value of the IDERIV-th derivative of the B-spline from the PP-representation.

Category F Solution of Nonlinear Equations

single double complx integr

CHKDER	DCKDER	-	Check the gradients of M nonlinear functions in N variables, evaluated at a point X, for consistency with the functions themselves.
FZERO	DFZERO	-	Find a zero of a function F(X) in a given interval [B,C]. It is designed primarily for problems where F(B) and F(C) have opposite signs.
RPQR79	-	CPQR79	Find the zeros of a polynomial with real (complex) coefficients using the companion matrix method.
RPZERO	-	CPZERO	Find the zeros of a polynomial with real (complex) coefficients using a modified Newton method.
SNSQ	DNSQ	-	Find a zero of a system of N nonlinear functions in N variables by a modification of the Powell hybrid method. This code is the combination of the MINPACK codes HYBRD and HYBRDJ.
SNSQE	DNSQE	-	The easy-to-use version of SNSQ. This code is the combination of the MINPACK codes HYBRD1 and HYBRJ1.

SOS DSOS - Solve a square system of nonlinear equations using a method similar to Brown's method.

Category G Optimization

single double complx integr

SPLP DSPLP - Solve linear programming problems involving at most a few thousand constraints and variables. Takes advantage of sparsity in the constraint matrix.

Category H2 Quadrature (Numerical Evaluation of Definite Integrals)

single double complx integr

AVINT DAVINT - Integrate a function tabulated at arbitrarily spaced abscissas using overlapping parabolas.

GAUS8 DGAUS8 - Integrate real functions of one variable over finite intervals using an adaptive 8-point Legendre-Gauss algorithm.

QAG DQAG - Estimate a definite integral over a finite interval.

QAGI DQAGI - Estimate a definite integral over a semi-infinite or infinite interval.

QAGP DQAGP - Estimate a definite integral over a finite interval with user supplied break points.

QAGS DQAGS - Estimate a definite integral over a finite interval with extrapolation.

QAWC DQAWC - Estimate a definite integral over a finite interval with a Cauchy weight function.

QAWF DQAWF - Estimate a definite integral over a semi-infinite interval with a Fourier weight function.

QAWO DQAWO - Estimate a definite integral over a finite interval with an oscillatory weight function.

QAWS DQAWS - Estimate a definite integral over a finite interval with an algebraic or logarithmic weight function.

QNC79 DQNC79 - Integrate a user defined function by a 7-point adaptive Newton-Cotes quadrature rule.

QNG DQNG - Estimate a definite integral over a finite interval using a non-adaptive scheme.

QPDOC Documentation for the QUADPACK numerical quadrature package.

QAGE DQAGE - The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F \text{ over } (A,B)$, hopefully satisfying following claim for accuracy $\text{ABS}(I-\text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.

QAGSE DQAGSE - The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F \text{ over } (A,B)$, hopefully satisfying following claim for accuracy $\text{ABS}(I-\text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.

QK15 DQK15 - To compute $I = \text{Integral of } F \text{ over } (A,B)$, with error estimate $J = \text{integral of } \text{ABS}(F) \text{ over } (A,B)$

QK21 DQK21 - To compute $I = \text{Integral of } F \text{ over } (A,B)$, with error estimate $J = \text{Integral of } \text{ABS}(F) \text{ over } (A,B)$

QK31 DQK31 - To compute $I = \text{Integral of } F \text{ over } (A,B)$ with error estimate $J = \text{Integral of } \text{ABS}(F) \text{ over } (A,B)$

QK41 DQK41 - To compute $I = \text{Integral of } F \text{ over } (A,B)$, with error estimate $J = \text{Integral of } \text{ABS}(F) \text{ over } (A,B)$

QK51 DQK51 - To compute $I = \text{Integral of } F \text{ over } (A,B)$ with error estimate $J = \text{Integral of } \text{ABS}(F) \text{ over } (A,B)$

QK61 DQK61 - To compute $I = \text{Integral of } F \text{ over } (A,B)$ with error estimate $J = \text{Integral of } \text{DABS}(F) \text{ over } (A,B)$

QAGPE DQAGPE - Approximate a given definite integral $I = \text{Integral of } F \text{ over } (A,B)$, hopefully satisfying the accuracy claim: $\text{ABS}(I-\text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$. Break points of the integration interval, where local difficulties of the integrand may occur (e.g. singularities or discontinuities) are provided by the user.

- QAWCE DQAWCE - The routine calculates an approximation result to a CAUCHY PRINCIPAL VALUE $I = \text{Integral of } F*W \text{ over } (A,B)$ ($W(X) = 1/(X-C)$, $(C.NE.A, C.NE.B)$), hopefully satisfying following claim for accuracy $ABS(I-RESULT) .LE. MAX(EPSABS, EPSREL*ABS(I))$
- QAWOE DQAWOE - Calculate an approximation to a given definite integral $I = \text{Integral of } F(X)*W(X) \text{ over } (A,B)$, where $W(X) = \cos(\text{OMEGA}*X)$ or $W(X)=\sin(\text{OMEGA}*X)$, hopefully satisfying the following claim for accuracy $ABS(I-RESULT) .LE. MAX(EPSABS, EPSREL*ABS(I))$.
- QAWSE DQAWSE - The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F*W \text{ over } (A,B)$, (where W shows a singular behaviour at the end points, see parameter INTEGR) hopefully satisfying following claim for accuracy $ABS(I-RESULT) .LE. MAX(EPSABS, EPSREL*ABS(I))$.
- QMOMO DQMOMO - This routine computes modified CHEBSYSHEV moments. The K -th modified CHEBYSHEV moment is defined as the integral over $(-1,1)$ of $W(X)*T(K,X)$, where $T(K,X)$ is the CHEBYSHEV POLYNOMIAL of degree K .
- QC25C DQC25C - To compute $I = \text{Integral of } F*W \text{ over } (A,B)$ with error estimate, where $W(X) = 1/(X-C)$
- QC25F DQC25F - To compute the integral $I=\text{Integral of } F(X) \text{ over } (A,B)$ Where $W(X) = \cos(\text{OMEGA}*X)$ or $W(X)=\sin(\text{OMEGA}*X)$ and to compute $J = \text{Integral of } ABS(F) \text{ over } (A,B)$. For small value of OMEGA or small intervals (A,B) the 15-point GAUSS-KRONRO rule is used. Otherwise a generalized CLENSHAW-CURTIS method is used.
- QC25S DQC25S - To compute $I = \text{Integral of } F*W \text{ over } (BL, BR)$, with error estimate, where the weight function W has a singular behaviour of ALGEBRAICO-LOGARITHMIC type at the points A and/or B . (BL, BR) is a part of (A,B) .
- QK15W DQK15W - To compute $I = \text{Integral of } F*W \text{ over } (A,B)$, with error estimate $J = \text{Integral of } ABS(F*W) \text{ over } (A,B)$
- QAGIE DQAGIE - The routine calculates an approximation result to a given integral $I = \text{Integral of } F \text{ over } (\text{BOUND}, +\text{INFINITY})$ or $I = \text{Integral of } F \text{ over } (-\text{INFINITY}, \text{BOUND})$ or $I = \text{Integral of } F \text{ over } (-\text{INFINITY}, +\text{INFINITY})$, hopefully satisfying following claim for accuracy $ABS(I-RESULT) .LE. MAX(EPSABS, EPSREL*ABS(I))$

- QAWFE DQAWFE - The routine calculates an approximation result to a given Fourier integral
I = Integral of F(X)*W(X) over (A, INFINITY)
where W(X)=COS(OMEGA*X) or W(X)=SIN(OMEGA*X),
hopefully satisfying following claim for accuracy
ABS(I-RESULT).LE.EPSABS.
- QK15I DQK15I - The original (infinite integration range is mapped onto the interval (0,1) and (A,B) is a part of (0,1). it is the purpose to compute
I = Integral of transformed integrand over (A,B),
J = Integral of ABS(Transformed Integrand) over (A,B).

Category I1 Ordinary Differential Equations

single double complx integr

- BVSUP DBVSUP - Solve a linear two-point boundary value problem using superposition coupled with an orthonormalization procedure and a variable-step integration scheme.
- DERKF DDERKF - Solve initial value problems in ordinary differential equations by the Runge-Kutta-Fehlberg fifth order method. Uses a variable step.
- DEABM DDEABM - Solve initial value problems in ordinary differential equations by an Adams method. Uses both variable (1-12) order and variable step.
- DEBDF DDEBDF - Solve stiff initial value problems in ordinary differential equations using backward differentiation formula. Uses both variable (1-5) order and variable step.
- SINTRP DINTP - Approximate the solution at XOUT by evaluating the polynomial computed in STEPS or DSTEPS.
- STEPS DSTEPS - Integrate a system of first order ODEs one step.

Category I2 Partial Differential Equations

single double complx integr

- BLKTRI - CBLKTR Solve a block tridiagonal system of linear equations (usually resulting from the discretization of separable two-dimensional elliptic equations).

-	-	CMGNBN	Solve a complex block tridiagonal linear system of equations by a cyclic reduction algorithm .
GENBUN	-	-	Solve using a cyclic reduction algorithm the linear system that results from a finite difference approximation to certain elliptic PDE's on a centered grid.
HSTCRT	-	-	Solve the standard five-point finite difference approximation on a staggered grid to the Helmholtz equation in Cartesian coordinates.
HSTCSP	-	-	Solve the standard five-point finite difference approximation on a staggered grid to the modified Helmholtz equation in spherical coordinates assuming axisymmetry (no dependence on longitude).
HSTCYL	-	-	Solve the standard five-point finite difference approximation on a staggered grid to the modified Helmholtz equation in cylindrical coordinates.
HSTPLR	-	-	Solve the standard five-point finite difference approximation on a staggered grid to the Helmholtz equation in polar coordinates.
HSTSSP	-	-	Solve the standard five-point finite difference approximation on a staggered grid to the Helmholtz equation in spherical coordinates and on the surface of the unit sphere (radius of 1).
HW3CRT	-	-	Solve the standard seven-point finite difference approximation to the Helmholtz equation in Cartesian coordinates.
HWSCRT	-	-	Solve the standard five-point finite difference approximation to the Helmholtz equation in Cartesian coordinates.
HWSCSP	-	-	Solve a finite difference approximation to the modified Helmholtz equation in spherical coordinates assuming axisymmetry (no dependence on longitude).
HWSCYL	-	-	Solve a standard finite difference approximation to the Helmholtz equation in cylindrical coordinates.
HWSPLR	-	-	Solve a finite difference approximation to the Helmholtz equation in polar coordinates.
HWSSSP	-	-	Solve a finite difference approximation to the Helmholtz equation in spherical coordinates and on the surface of

the unit sphere (radius of 1).

- POIS3D - - Solve three-dimensional block tridiagonal linear systems arising from finite difference approximations to three-dimensional Poisson equations using the Fourier transform package written by Paul Swarztrauber.
- POISTG - - Solve a block tridiagonal system of linear equations that results from a staggered grid finite difference approximation to 2-d elliptic PDE's.
- SEPELI - - Automatically discretizes and solves second and fourth (optionally) order finite difference approximations on a uniform grid to the general separable elliptic PDE on a rectangle with any combination of periodic or mixed boundary conditions.
- SEPX4 - - Solve for either the second or fourth order finite difference approximation to the solution of a separable elliptic equation on a rectangle. Any combination of periodic or mixed boundary conditions is allowed.
 SEPX4 is 3 times faster than SEPELI

Category J1 Fast Fourier Transform

single double complx integr

- COSQB - - Inverse cosine transform with odd wave numbers.
 The unnormalized inverse of COSQF.
- COSQF - - Forward cosine transform with odd wave numbers.
- COSQI - - Initialize for COSQF and COSQB.
- COST - - Cosine transform of a real, even sequence.
- COSTI - - Initialize for COST.
- EZFFTB - - Simplified real, periodic, inverse (backward) transform.
- EZFFTF - - Simplified real, periodic, forward transform.
- EZFFTI - - Initialize for EZFFTF and EZFFTB.
- RFFTB - CFFTB Inverse (backward) transform of a real (complex)
 periodic coefficient array.

RFFTF - CFFTF Forward transform of a real (complex), periodic sequence.

RFFTI - CFFTI Initialize for *FFTF and *FFTB.

SINQB - - Inverse sine transform with odd wave numbers.
The unnormalized inverse of SINQF.

SINQF - - Forward sine transform with odd wave numbers.

SINQI - - Initialize for SINQF and SINQB.

SINT - - Sine transform of a real, odd sequence.

SINTI - - Initialize for SINT.

Category K Approximation

single double complx integr

EFC DEFC - Fit a piece-wise polynomial curve to discrete data.
The piece-wise polynomials are represented as B-splines.
The fitting is done in a weighted least squares sense.

FC DFC - Fit a piece-wise polynomial curve to discrete data.
The piece-wise polynomials are represented as B-splines.
The fitting is done in a least squares sense. Equality
and inequality constraints can be imposed on the fitted
curve.

POLFIT DPOLFT - Fit a least squares polynomial to discrete data in
one variable.

PVALUE DP1VLU - Use the coefficients generated by POLFIT to evaluate
the polynomial fit of degree L, along with the first
NDER of its derivatives, at a specified point.

SBOCLS DBOCLS - Solve the bounded and constrained least squares
problem $EX = F$ with linear constraints $CX = Y$ and
bounds on selected values of X and/or Y.

SBOLS DBOLS - Solve the bounded least squares problem $EX = F$
with bounds on selected values of X.

SCOV DCOV - Calculate the covariance matrix for a nonlinear data
fitting problem. It is intended to be used after a
successful return from either SNLS1 or SNLS1E.

SNLS1 DNLS1 - Minimize the sum of the squares of M nonlinear functions in N variables by a modification of the Levenberg-Marquardt algorithm. This code is a combination of the MINPACK codes LMDER, LMDIF, and LMSTR.

SNLS1E DNLS1E - The easy-to-use version of SNLS1. This code is a combination of the MINPACK codes LMDER1, LMDIF1, and LMSTR.

LSEI DLSEI - Solve a linearly constrained least squares problem with equality and inequality constraints, and optionally compute a covariance matrix.

WNNLS DNNLS - Solve a linearly constrained least squares problem with equality constraints and nonnegativity constraints on selected variables.

Category L6 Pseudo-Random Number Generation

single double complx integr

RAND - - Generate a uniformly distributed random number.

RGAUSS - - Generate a normally distributed (Gaussian) random number

RUNIF - - A portable uniform random number generator.

Category N6 Sorting

single double complx integr

SSORT DSORT - ISORT
Sort an array X and optionally make the same interchanges in array Y. A slightly modified QUICKSORT algorithm is used to sort the array in either increasing or decreasing order.

Category R1 Machine Constants

single double complx integr

R1MACH D1MACH - Return real (double) machine dependent constants.

- - - I1MACH

Return integer machine dependent constants.

Category R3 Diagnostics and Error Handling

FDUMP Symbolic dump (should be locally written).

XERABT Abort program execution and print error message.

XERCLR Reset the current error number to zero.

XERCTL Allows user control over handling of individual errors.

XERDMP Print the error tables and then clears them.

XERMAX Set maximum number of times any error message is to be printed.

XERPRT Print error messages.

XERROR Process an error (diagnostic) message.

XERRWV Process an error message allowing two integer and two real values to be included in the message.

XERSAV Record that an error occurred.

XGETF Return the current value of error control flag.

XGETUA Return the unit number(s) to which error messages are being sent.

XGETUN Return the (first) output file to which messages are being sent.

XSETF Set the error control flag.

XSETUA Set up to 5 unit numbers to which messages are to be sent.

XSETUN Set the output file to which error messages are to be sent.

Category Z Other

AAAAAA The SLATEC disclaimer.

BSPDOC Documentation for the B-spline routines.

EISDOC Documentation for the EISPACK eigenvalue package.

FFTDOC Documentation for the Fast Fourier Transform package.

FNLIBD Documentation for the Elementary and Special Functions.

MENU Contents of the SLATEC library listing subprogram names only.

PCHDOC Documentation for the PCHIP routines. PCHIP is a Fortran package for piecewise cubic Hermite interpolation of data. It features software to produce a monotone and "visually pleasing" interpolant to monotone data.

QPDOG Documentation for the QUADPACK numerical quadrature package.

*****LINPACK Routines*****

single double complx integr

SCHDC	DCHDC	CCHDC	Compute the Cholesky decomposition of a positive definite matrix. A pivoting option allows the user to estimate the condition or rank of the matrix.
SCHDD	DCHDD	CCHDD	Downdate an augmented Cholesky decomposition or the triangular factor of an augmented QR decomposition.
SCHEX	DCHEX	CCHEX	Update the Cholesky factorization $A = CTRANS(R)*R$ of a positive definite matrix A under diagonal permutations of the form $U*R*E=RR$, where E is a permutation matrix.
SCHUD	DCHUD	CCHUD	Update an augmented Cholesky decomposition of the triangular part of an augmented QR decomposition.

-----GENERAL BAND MATRICES-----

SGBCO	DGBCO	CGBCO	Factor (LU) a band matrix by Gaussian elimination and estimate the condition of the matrix.
SGBDI	DGBDI	CGBDI	Compute the determinant of a band matrix. Use N times for the inverse.
SGBFA	DGBFA	CGBFA	Factor (LU) a band matrix by elimination.
SGBSL	DGBSL	CGBSL	Solve the band system $A*X=B$ or $CTrans(A)*X=B$.

-----GENERAL MATRICES-----

SGECO DGECC CGECO Factor (LU) a matrix by Gaussian elimination and estimate the condition number.

SGEDI DGEDC CGEDI Compute the determinant and inverse of a matrix.

SGEFA DGEFA CGEFA Factor (LU) a matrix by Gaussian elimination.

SGESL DGESL CGESL Solve the system $A^*X=B$ or $TRANS(A)^*X=B$ using the factors computed by *GECO or *GEFA.

----GENERAL TRIDIAGONAL MATRICES----

SGTSL DGTSL CGTSL Solve the system $T^*X=B$ where T is a tridiagonal matrix.

----HERMITIAN POSITIVE DEFINITE BAND MATRICES----

SPBCO DPBCO CPBCO Factor (LU) a Hermitian positive definite matrix stored in band form and estimate the condition of the matrix.

SPBDI DPBDI CPBDI Compute the determinant of a Hermitian positive definite band matrix using factors from *PBCO or *PBFA. For the inverse use *PBSL.

SPBFA DPBFA CPBFA Factor (LU) a Hermitian positive definite matrix stored in band form.

SPBSL DPBSL CPBSL Solve the Hermitian positive definite band system $A^*X=B$ using factors from *PBCO or *PBFA.

----HERMITIAN POSITIVE DEFINITE MATRICES----

SPOCO DPOCO CPOCO Factor (LU) a Hermitian positive definite matrix and estimate the condition of the matrix.

SPODI DPODI CPODI Compute the determinant and inverse of a Hermitian positive definite matrix using factors of *POCO, *POFA or *QRDC.

SPOFA DPOFA CPOFA Factor (LU) a Hermitian positive definite matrix.

SPOSL DPOSL CPOSL Solve the Hermitian positive definite system $A^*X=B$ using the factors computed by *POCO or *POFA.

----HERMITIAN POSITIVE DEFINITE MATRICES (PACKED FORM)----

SPPCO DPPCO CPPCO Factor (LU) a Hermitian positive definite matrix stored in packed form and estimate the condition of the matrix.

SPPDI DPPDI CPPDI Compute the determinant and inverse of a Hermitian positive definite matrix using factors from *PPCO or *PPFA

SPPFA DPPFA CPPFA Factor (LU) a Hermitian positive definite matrix stored in packed form.

SPPSL DPPSL CPPSL Solve the Hermitian positive definite system $A^*X=B$ using the factors computed by *PPCO or *PPFA.

----POSITIVE DEFINITE TRIDIAGONAL MATRICES----

SPTSL DPTSL CPTSL Solve a positive definite tridiagonal system.

---SYMMETRIC MATRICES----

SSICO DSICO CSICO Factor (LU) a symmetric matrix by elimination with symmetric pivoting and estimate the condition number.

SSIDI DSIDI CSIDI Compute the determinant and inverse of a symmetric matrix using the factors from *SIFA.

SSIFA DSIFA CSIFA Factor (LU) a symmetric matrix by elimination with symmetric pivoting.

SSISL DSISL CSISL Solve the symmetric system $A^*X=B$ using factors from *SIFA.

---SYMMETRIC MATRICES (PACKED FORM)----

SSPCO DSPCO CSPCO Factor (LU) a symmetric matrix stored in packed form by elimination with symmetric pivoting and estimate the condition of the matrix.

SSPDI DSPDI CSPDI Compute the determinant and inverse of a symmetric matrix stored in packed form using factors from *SPFA.

SSPFA DSPFA CSPFA Factor (LU) a symmetric matrix stored in packed form by elimination with symmetric pivoting.

SSPSL DSPSL CSPSL Solve the symmetric system $A^*X=B$ using factors from *SPFA.

---TRIANGULAR MATRICES----

STRCO DTRCO CTRCO Estimates the condition of a triangular system.

STRDI DTRDI CTRDI Compute the determinant and inverse of a triangular matrix.

STRSL DTRSL CTRSL Solve systems of the form $T^*X=B$ or $TRANS(T)^*X=B$
where T is a triangular matrix.

---COMPLEX HERMITIAN MATRICES----

- - CHICO Factor (LU) a complex Hermitian matrix by elimination with symmetric pivoting and estimate the condition number.
- - CHIDI Compute the determinant, inertia and inverse of a complex Hermitian matrix using the factors from CHIFA.
- - CHIFA Factor (LU) a Hermitian matrix by elimination with symmetric pivoting.
- - CHISL Solve the Hermitian system $A^*X=B$ using factors of CHIFA.

---COMPLEX HERMITIAN MATRICES (PACKED FORM)----

- - CHPCO Factor (LU) a complex Hermitian matrix (packed form) by elimination with symmetric pivoting and estimate the condition of the matrix.
- - CHPDI Compute the determinant, inertia and inverse of a complex Hermitian matrix (packed form) using the factors from CHPFA.
- - CHPFA Factor (LU) a complex Hermitian matrix (packed form) by elimination with symmetric pivoting.
- - CHPSL Solve the Hermitian system $A^*X=B$ using factors of CHPFA.

*****EISPACK Routines*****

single double complx integr

- RS - CH Computes eigenvalues and, optionally, eigenvectors of real symmetric (complex Hermitian) matrix.
- RSP - - Compute eigenvalues and, optionally, eigenvectors of real symmetric matrix packed into a one dimensional array.
- RG - CG Computes eigenvalues and, optionally, eigenvectors of a real (complex) general matrix.

RSGAB	-	-	Computes eigenvalues and, optionally, eigenvectors of symmetric generalized eigenproblem: $A*B*X=(LAMBDA)*X$
RSGBA	-	-	Computes eigenvalues and, optionally, eigenvectors of symmetric generalized eigenproblem: $B*A*X=(LAMBDA)*X$
RGG	-	-	Computes eigenvalues and eigenvectors for real generalized eigenproblem: $A*X=(LAMBDA)*B*X$.
BALANC	-	CBAL	Balances a general real (complex) matrix and isolates eigenvalues whenever possible.
BANDR	-	-	Reduces real symmetric band matrix to symmetric tridiagonal matrix and, optionally, accumulates orthogonal similarity transformations.
HTRID3	-	-	Reduces complex Hermitian (packed) matrix to real symmetric tridiagonal matrix by unitary similarity transformations.
HTRIDI	-	-	Reduces complex Hermitian matrix to real symmetric tridiagonal matrix using unitary similarity transformations.
TRED1	-	-	Reduce real symmetric matrix to symmetric tridiagonal matrix using orthogonal similarity transformations.
TRED2	-	-	Reduce real symmetric matrix to symmetric tridiagonal matrix using and accumulating orthogonal transformations.
TRED3	-	-	Reduce real symmetric matrix stored in packed form to symmetric tridiagonal matrix using orthogonal transformations.
ELMHES	-	COMHES	Reduces real (complex) general matrix to upper Hessenberg form using stabilized elementary similarity transformations.
ORTHES	-	CORTH	Reduces real (complex) general matrix to upper Hessenberg form orthogonal (unitary) similarity transformations.
QZHES	-	-	The 1st step of the QZ algorithm for solving generalized matrix eigenproblems. Accepts a pair of real general matrices and reduces one of them to upper Hessenberg and the other to upper triangular form using orthogonal transformations. Usually followed by QZIT, QZVAL, QZ
QZIT	-	-	The second step of the QZ algorithm for generalized

eigenproblems. Accepts an upper Hessenberg and an upper triangular matrix and reduces the former to quasi-triangular form while preserving the form of the latter. Usually preceded by QZHES and followed by QZVAL and QZVEC.

FIGI	-	-	Transforms certain real non-symmetric tridiagonal matrix to symmetric tridiagonal matrix.
FIGI2	-	-	Transforms certain real non-symmetric tridiagonal matrix to symmetric tridiagonal matrix.
REDUC	-	-	Reduces generalized symmetric eigenproblem $A*X=(LAMBDA)*B*X$, to standard symmetric eigenproblem using Cholesky factorization.
REDUC2	-	-	Reduces certain generalized symmetric eigenproblems standard symmetric eigenproblem, using Cholesky factorization.
-	-	COMLR	Computes eigenvalues of a complex upper Hessenberg m using the modified LR method.
-	-	COMLR2	Computes eigenvalues and eigenvectors of complex upper Hessenberg matrix using modified LR method.
HQR	-	COMQR	Computes eigenvalues of a real (complex) upper Hessenberg matrix using the QR method.
HQR2	-	COMQR2	Computes eigenvalues and eigenvectors of real (complex) upper Hessenberg matrix using QR method.
INVIT	-	CINVIT	Computes eigenvectors of real (complex) Hessenberg matrix associated with specified eigenvalues by inverse iteration.
QZVAL	-	-	The third step of the QZ algorithm for generalized eigenproblems. Accepts a pair of real matrices, one quasi-triangular form and the other in upper triangular form and computes the eigenvalues of the associated eigenproblem. Usually preceded by QZHES, QZIT, and followed by QZVEC.
BANDV	-	-	Forms eigenvectors of real symmetric band matrix associated with a set of ordered approximate eigenvalue by inverse iteration.
QZVEC	-	-	The optional fourth step of the QZ algorithm for generalized eigenproblems. Accepts a matrix in

quasi-triangular form and another in upper triangular and computes the eigenvectors of the triangular problem and transforms them back to the original coordinates Usually preceded by QZHES, QZIT, QZVAL.

TINVIT	-	-	Eigenvectors of symmetric tridiagonal matrix corresponding to some specified eigenvalues, using inverse iteration.
BAKVEC	-	-	Forms eigenvectors of certain real non-symmetric tridiagonal matrix from symmetric tridiagonal matrix output from FIGI.
BALBAK	-	CBABK2	Forms eigenvectors of real (complex) general matrix from eigenvectors of matrix output from BALANC (CBAL).
ELMBAK	-	COMBAK	Forms eigenvectors of real (complex) general matrix from eigenvectors of upper Hessenberg matrix output from ELMHES (COMHES).
ELTRAN	-	-	Accumulates the stabilized elementary similarity transformations used in the reduction of a real general matrix to upper Hessenberg form by ELMHES.
HTRIB3	-	-	Computes eigenvectors of complex Hermitian matrix from eigenvectors of real symmetric tridiagonal matrix output from HTRID3.
HTRIBK	-	-	Forms eigenvectors of complex Hermitian matrix from eigenvectors of real symmetric tridiagonal matrix output from HTRIDI.
ORTBAK	-	CORTB	Forms eigenvectors of general real (complex) matrix from eigenvectors of upper Hessenberg matrix output from ORTHES (CORTH).
ORTRAN	-	-	Accumulates orthogonal similarity transformations in reduction of real general matrix by ORTHES.
REBAK	-	-	Forms eigenvectors of generalized symmetric eigensystem from eigenvectors of derived matrix output from REDUC REDUC2.
REBAKB	-	-	Forms eigenvectors of generalized symmetric eigensystem from eigenvectors of derived matrix output from REDUC2
TRBAK1	-	-	Forms the eigenvectors of real symmetric matrix from eigenvectors of symmetric tridiagonal matrix formed by TRED1.

TRBAK3 - - Forms eigenvectors of real symmetric matrix from the
 eigenvectors of symmetric tridiagonal matrix formed by
 TRED3.

MINFIT - - Compute Singular Value Decomposition of rectangular
 matrix and solve related Linear Least Squares problem.

A4A	CARG	Computes the argument of a complex number.
A6B	D9PAK	Pack a base 2 exponent into a double precision floating point number.
A6B	D9UPAK	Unpack a double precision floating point no. x so that $X=Y*2^{**N}$.
A6B	R9PAK	Pack a base 2 exponent into a floating point number.
A6B	R9UPAK	Unpack a floating point number X so that $X=Y*2^{**N}$.
C	FNLIBD	Documentation for Elementary and Special Functions
C1	BINOM	Computes the binomial coefficients.
C1	DBINOM	Computes the d.p. binomial coefficients.
C1	DFAC	Computes the d.p. factorial of N.
C1	DPOCH	Computes d.p. generalized Pochhammer's symbol.
C1	DPOCH1	Calculates a generalization of Pochhammer's symbol starting from first order. (double precision)
C1	FAC	Computes the factorial of N.
C1	POCH	Evaluates a generalization of Pochhammer's symbol.
C1	POCH1	Computes Pochhammer's symbol from first order.
C2	CBRT	Computes the cube root of X.
C2	CCBRT	Computes the complex cube root of Z.
C2	DCBRT	Computes the d.p. cube root.
C3A2	CSEVL	Evaluate the N-term Chebyshev series CS at X.
C3A2	DCSEVL	Evaluate the double precision N-term Chebyshev series A at X.
C3A2	INITDS	Initializes the d.p. properly normalized orthogonal polynomial series to determine the number of terms needed for specific accuracy.
C3A2	INITS	Initializes an orthogonal series so that it defines the number of terms to carry in the series to meet a specified error.

C4A	CACOS	Computes the complex arc Cosine.
C4A	CASIN	Computes the complex arc Sine.
C4A	CATAN	Computes the complex arc Tangent.
C4A	CATAN2	Computes the complex arc Tangent in the proper quadrant.
C4A	CCOT	Computes the complex Cotangent.
C4A	COSDG	Computes the Cosine of an argument in degrees.
C4A	COT	Computes the Cotangent.
C4A	CTAN	Computes the complex Tangent.
C4A	D9ATN1	Evaluates DATAN(X) from first order relative accuracy.
C4A	DCOSDG	Computes the d.p. Cosine for degree arguments.
C4A	DCOT	Computes the d.p. Cotangent.
C4A	DSINDG	Computes the d.p. Sine for degree arguments.
C4A	R9ATN1	Evaluate ATAN(X) from first order relative accuracy so that $ATAN(X) = X + X**3*R9ATN1(X)$.
C4A	SINDG	Computes the Sine of an argument in degrees.
C4B	ALNREL	Evaluates $\ln(1+X)$ accurate in the sense of relative error.
C4B	C9LN2R	Evaluates $CLOG(1+Z)$ from second order with relative error so that $CLOG(1+Z) = Z - Z**2/2 + Z**3*C9LN2R(Z)$.
C4B	CEXPRL	Evaluates $(CEXP(Z)-1)/Z$ so that $EXP(Z) = 1 + Z*CEXPRL(Z)$
C4B	CLNREL	Computes the principal value of the complex natural logarithm of $1+Z$ with relative error accuracy for small $CABS(Z)$.
C4B	CLOG10	Computes the principal value of the complex base 10 logarithm.
C4B	D9LN2R	Evaluate $DLOG(1+X)$ from second order relative accuracy so that $DLOG(1+X) = X - X**2/2 + X**3*D9LN2R(X)$
C4B	DEXPRL	Calculates the d.p. relative error exponential $(DEXP(X)-1)/X$.

C4B	DLNREL	Evaluates d.p. $\text{LN}(1+X)$ accurate in the relative error sense
C4B	EXPREL	Evaluates $\text{EXPREL}(X) = (\text{EXP}(X) - 1)/X$
C4B	R9LN2R	Evaluates $\text{ALOG}(1+X)$ from second order relative accuracy so that $\text{ALOG}(1+X) = X - X^2/2 + X^3 * \text{R9LN2R}(X)$.
C4C	ACOSH	Compute the arc hyperbolic Cosine.
C4C	ASINH	Computes the arc hyperbolic Sine.
C4C	ATANH	Compute the arc hyperbolic Tangent.
C4C	CACOSH	Computes the complex arc hyperbolic Cosine.
C4C	CASINH	Computes the complex arc hyperbolic Sine.
C4C	CATANH	Computes the complex arc hyperbolic Tangent.
C4C	CCOSH	Computes the complex hyperbolic Cosine.
C4C	CSINH	Computes the complex hyperbolic Sine.
C4C	CTANH	Computes the complex hyperbolic Tangent.
C4C	DACOSH	Computes the arc hyperbolic Cosine.
C4C	DASINH	Computes the arc hyperbolic Sine.
C4C	DATANH	Computes the d.p. arc hyperbolic Tangent.
C5	ALI	Computes the logarithmic integral.
C5	DEI	Computes the d.p. exponential integral $E_1(X)$.
C5	DEI	Computes the d.p. exponential integral $E_1(X)$.
C5	DEXINT	DEXINT computes M member sequences of exponential integrals $E(N+K, X)$, $K=0, 1, \dots, M-1$ for $N \geq 1$ and $X \geq 0$.
C5	DLI	Computes the d.p. logarithmic integral.
C5	DSPENC	Computes a d.p. form of Spence's integral due to K. Mitchell.
C5	E1	Computes the exponential integral $E_1(X)$.
C5	EI	Computes the exponential integral $E_1(X)$.

C5	EXINT	EXINT computes M member sequences of exponential integrals $E(N+K, X)$, $K=0, 1, \dots, M-1$ for $N \geq 1$ and $X \geq 0$.
C5	SPENC	Compute Spence's function.
C7A	ALGAMS	Computes $\log(\text{ABS}(\text{GAMMA}(X)))$.
C7A	ALNGAM	Computes the log of the absolute value of the Gamma function
C7A	COLGMC	Evaluates $(Z+0.5)*\text{CLOG}((Z+1.)/Z) - 1.0$ with relative accuracy.
C7A	C9LGMC	Computes the LOG GAMMA correction term for most Z so that $\text{CLOG}(\text{CGAMMA}(Z)) = 0.5*\text{ALOG}(2.*\text{PI}) + (Z-0.5)*\text{CLOG}(Z) - Z + \text{C9LGMC}(Z)$
C7A	CGAMMA	Computes the Gamma function of complex argument.
C7A	CGAMR	Computes the reciprocal Gamma function of complex argument.
C7A	CLNGAM	CLNGAM computes the natural log of the complex valued Gamma function at ZIN, where ZIN is a complex number.
C7A	DGAMLM	Computes the d.p. minimum and maximum bounds for X in $\text{GAMMA}(X)$.
C7A	DGAMMA	Computes the d.p. complete Gamma function.
C7A	DGAMR	Calculates d.p. reciprocal Gamma function.
C7A	DLGAMS	Calculates the log of the absolute value of the Gamma function
C7A	DLNGAM	Computes the d.p. logarithm of the absolute value of the Gamma function
C7A	GAMLM	Computes the minimum and maximum bounds for X in $\text{GAMMA}(X)$.
C7A	GAMMA	Computes the Gamma function.
C7A	GAMR	Computes the reciprocal of the Gamma function.
C7B	ALBETA	Computes the natural log of the complete Beta function.
C7B	BETA	Computes the complete Beta function.

C7B	CBETA	CBETA computes the complete Beta function of complex parameters A and B.
C7B	CLBETA	CLBETA computes the natural log of the complex valued complete Beta function of complex parameters A and B.
C7B	DBETA	Computes the d.p. complete Beta function.
C7B	DLBETA	Computes the d.p. natural logarithm of the complete Beta function.
C7C	CPSI	Computes the Psi function of complex argument.
C7C	DPSI	Computes the d.p. Psi (or Digamma) function.
C7C	PSI	Computes the Psi (or Digamma) function.
C7E	D9GMIC	Calculates the d.p. incomplete Gamma function for A near a negative integer and X small.
C7E	D9GMIT	Computes d.p. Tricomi's incomplete Gamma function for small X.
C7E	D9LGIC	Computes the d.p. log incomplete Gamma function for large X and for A .LE. X.
C7E	D9LGIT	Computes the log of Tricomi's incomplete Gamma function with Perron's continued fraction for large X and A .GE. X.
C7E	D9LGMC	Computes the d.p. log Gamma correction factor for X .GE. 10. so that $DLOG(DGAMMA(X)) = DLOG(DSQRT(2*PI)) + (X-5.)*DLOG(X) - X + D9LGMC(X)$
C7E	DGAMI	Calculates the d.p. incomplete Gamma function.
C7E	DGAMIC	Calculates the d.p. complementary incomplete Gamma function
C7E	DGAMIT	Calculates Tricomi's form of the incomplete Gamma function.
C7E	GAMI	Computes the incomplete Gamma function.
C7E	GAMIC	Computes the complementary incomplete Gamma function.
C7E	GAMIT	Computes Tricomi's form of the incomplete Gamma function.

C7E	R9GMIC	Compute the complementary incomplete Gamma function for A near a negative integer and for small X.
C7E	R9GMIT	Computes Tricomi's incomplete Gamma function for small X.
C7E	R9LGIC	Computes the log complementary incomplete Gamma function for large X and for A .LE. X.
C7E	R9LGIT	Computes the log of Tricomi's incomplete Gamma function with Perron's continued fraction for large X and A .GE. X.
C7E	R9LGMC	Computes the log Gamma correction factor so that $\text{ALOG}(\text{GAMMA}(X)) = \text{ALOG}(\text{SQRT}(2*\text{PI})) + (X-.5)*\text{ALOG}(X) - X + \text{R9LGMC}(X)$
C7F	BETAI	Computes the incomplete Beta function.
C7F	DBETAI	Calculates the d.p. incomplete Beta function.
C8A	DERF	Computes the d.p. error function, ERF, of X.
C8A	DERFC	Computes the d.p. complementary error function, ERFC.
C8A	ERF	Computes the error function, ERF.
C8A	ERFC	Computes the complementary error function (ERFC).
C8C	DAWS	Computes Dawson's function.
C8C	DDAWS	Computes the d.p. Dawson's function.
C10A1	BESJ0	Computes the Bessel function of the first kind of order zero.
C10A1	BESJ1	Computes the Bessel function of the first kind of order one.
C10A1	BESY0	Computes the Bessel function of the second kind of order zero.
C10A1	BESY1	Computes the Bessel function of the second kind of order one.
C10A1	D9B0MP	Evaluates the d.p. modulus and phase for J0 and Y0 Bessel functions.
C10A1	D9B1MP	Evaluate the d.p. modulus and phase for the J1 and Y1 Bessel functions.

C10A1	DBESJ0	Computes the d.p. Bessel function of the first kind of order zero.
C10A1	DBESJ1	Computes the d.p. Bessel function of the first kind of order one.
C10A1	DBESY0	Computes the d.p. Bessel function of the second kind of order zero.
C10A1	DBESY1	Computes the d.p. Bessel function of the second kind of order one.
C10A3	BESJ	BESJ computes an N member sequence of J Bessel functions $J/\text{SUB}(\text{ALPHA}+K-1)/(X)$, $K=1,\dots,N$ for non-negative ALPHA and X
C10A3	BESY	BESY implements forward recursion on the three term recursion relation for a sequence of non-negative order Bessel functions $Y/\text{SUB}(\text{FNU}+I-1)/(X)$, $I=1,N$ for real $X.GT.0.0E0$ and non-negative orders FNU.
C10A3	DBESJ	DBESJ computes an N member sequence of J Bessel functions $J/\text{SUB}(\text{ALPHA}+K-1)/(X)$, $K=1,\dots,N$ for non-negative ALPHA and X.
C10A3	DBESY	DBESY implements forward recursion on the three term recursion relation for a sequence of non-negative order Bessel functions $Y/\text{SUB}(\text{FNU}+I-1)/(X)$, $I=1,N$ for real $X.GT.0.0D0$ and non-negative orders FNU.
C10B1	BESI0	Computes the hyperbolic Bessel function of the first kind of order zero.
C10B1	BESI0E	Computes the exponentially scaled hyperbolic Bessel function of the first kind of order zero.
C10B1	BESI1	Computes the hyperbolic Bessel function of first kind of order one.
C10B1	BESI1E	Computes the exponentially scaled hyperbolic Bessel function of the first kind of order one.
C10B1	BESK0	Computes the hyperbolic Bessel function of the third kind of order zero.
C10B1	BESK0E	Computes the exponentially scaled hyperbolic Bessel function of the third kind of order zero.

C10B1	BESK1	Computes the hyperbolic Bessel function of the third kind of order one.
C10B1	BESK1E	Computes the exponentially scaled hyperbolic Bessel function of the third kind of order one.
C10B1	DBES10	Computes the d.p. hyperbolic Bessel function of the first kind of order zero.
C10B1	DBES11	Computes the d.p. modified (hyperbolic) Bessel function of the first kind of order one.
C10B1	DBESK0	Computes d.p. modified (hyperbolic) Bessel function of the third kind of order zero.
C10B1	DBESK1	Computes the dp modified Bessel function of the third kind of order one.
C10B1	DBS10E	Computes the d.p. exponentially scaled hyperbolic Bessel function of the first kind of order zero.
C10B1	DBS11E	Computes the d.p. exponentially scaled modified (hyperbolic) Bessel function of the first kind of order one.
C10B1	DBSK0E	Computes the d.p. exponentially scaled modified (hyperbolic) Bessel function of the third kind of order zero.
C10B1	DBSK1E	Computes the exponentially scaled, modified hyperbolic Bessel function of the third kind of order one (double precision).
C10B3	BESI	BESI computes an N member sequence of I Bessel functions $I/\text{SUB}(\text{ALPHA}+K-1)/(X)$, $K=1,\dots,N$ or scaled Bessel functions $\text{EXP}(-X)*I/\text{SUB}(\text{ALPHA}+K-1)/(X)$, $K=1,\dots,N$ for non-negative ALPHA and X.
C10B3	BESK	BESK implements forward recursion on the three term recursion relation for a sequence of non-negative order Bessel functions $K/\text{SUB}(\text{FNU}+I-1)/(X)$, or scaled Bessel functions $\text{EXP}(X)*K/\text{SUB}(\text{FNU}+I-1)/(X)$, $I=1,\dots,N$ for real $X.GT.0.0E0$ and non-negative orders FNU.
C10B3	BESKES	Computes a sequence of exponentially scaled modified Bessel functions of the third kind of fractional order.
C10B3	BESKS	Computes a sequence of modified Bessel functions of the third kind of fractional order.

C10B3	D9KNUS	Computes Bessel functions $\text{EXP}(X) * K - \text{SUB-XNU}(X)$ and $\text{EXP}(X) * K - \text{SUB-XNU} + 1(X)$ for $0. \leq XNU < 1.$
C10B3	DBESI	DBESI computes an N member sequence of I Bessel functions $I / \text{SUB}(\text{ALPHA} + K - 1) / (X), K = 1, \dots, N$ or scaled Bessel functions $\text{DEXP}(-X) * I / \text{SUB}(\text{ALPHA} + K - 1) / (X), K = 1, \dots, N$ for nonnegative ALPHA and X.
C10B3	DBESK	DBESK implements forward recursion on the three term recursion relation for a sequence of non-negative order Bessel functions $K / \text{SUB}(\text{FNU} + I - 1) / (X),$ or scaled Bessel functions $\text{DEXP}(X) * K / \text{SUB}(\text{FNU} + I - 1) / (X), I = 1, \dots, N$ for real $X > 0.0D0$ and non-negative orders FNU.
C10B3	DBESKS	Computes a d.p. sequence of modified Bessel functions of the third kind of fractional order.
C10B3	DBSKES	Computes a d.p. sequence of exponentially scaled modified Bessel functions of the third kind of fractional order.
C10B3	R9KNUS	Computes Bessel functions $\text{EXP}(X) * K - \text{SUB-XNU}(X)$ and $\text{EXP}(X) * K - \text{SUB-XNU} + 1(X)$ for $0.0 \leq XNU < 1.0.$
C10D	AI	Computes the Airy function.
C10D	AIE	Computes the exponentially scaled Airy function.
C10D	BI	Computes the Bairy function.
C10D	BIE	Computes the exponentially scaled Bairy function.
C10D	D9AIMF	Evaluates the Airy modulus and phase for $X \leq -1.0$
C10D	DAI	Calculates the d.p. Airy function.
C10D	DAIE	Calculates the d.p. Airy function for negative X and an exponentially scaled Airy function for positive X.
C10D	DBI	Calculates the d.p. Bairy function (Airy of second kind).
C10D	DBIE	Calculates the d.p. Bairy function for negative X and an exponentially scaled Bairy function for positive X.
C10D	R9AIMF	Evaluate the Airy modulus and phase.
C11	CHU	Computes the logarithmic confluent hypergeometric function.

C11	D9CHU	Evaluates for large Z $Z^{**A} * U(A,B,Z)$ where U is the logarithmic confluent hypergeometric function. (double precision)
C11	DCHU	Calculates the d.p. logarithmic confluent hyper-geometric function.
C11	R9CHU	Evaluates for large Z $Z^{**A} * CHU(A,B,Z)$.
D1A2	ICAMAX	Find largest component of complex vector.
D1A2	IDAMAX	Find largest component of d.p. vector.
D1A2	ISAMAX	Find largest component of s.p. vector.
D1A3A	DASUM	Sum of magnitudes of d.p. vector components.
D1A3A	SASUM	Sum of magnitudes of s.p vector components.
D1A3A	SCASUM	Sum of magnitudes of real and imaginary components of complex vector.
D1A3B	DNRM2	Euclidean length (L2 norm) of d.p. vector.
D1A3B	SCNRM2	Unitary norm of complex vector.
D1A3B	SNRM2	Euclidean length (L2 norm) of s.p. vector.
D1A4	CDCDOT	Complex inner product with result accumulated in d.p.
D1A4	CDOTC	Dot product of complex vectors, uses complex conjugate of first vector.
D1A4	CDOTU	Inner product of complex vectors.
D1A4	DCDOT	Computes the dot product of 2 complex vectors, CX and CY, e.g. CX DOT CY, or, CXconjugate DOT CY. The real and imaginary parts of CX and CY are converted to double precision, the dot product accumulation is done in double precision and the output is given as 2 double precision numbers, corresponding to the real and imaginary part of the result.
D1A4	DDOT	D.P. inner product of d.p. vectors.
D1A4	DQDOTA	D.P. inner product with extended precision accumulation and result.

D1A4	DQDOTI	D.P. inner product with extended precision accumulation and result.
D1A4	DSDOT	D.P inner product of s.p. vectors.
D1A4	SDOT	S.P. inner product of s.p. vectors.
D1A4	SDSDOT	S.P. result with inner product accumulated in d.p.
D1A5	CCOPY	Complex vector copy $y = x$.
D1A5	CSWAP	Interchange complex vectors.
D1A5	DCOPY	D.P. vector copy $y = x$.
D1A5	DSWAP	Interchange d.p. vectors.
D1A5	SCOPY	Copy s.p. vector $y = x$.
D1A5	SCOPYM	Copy negative of real SX to real SY.
D1A5	SSWAP	Interchange s.p vectors.
D1A6	CSCAL	Complex vector scale $x = a*x$.
D1A6	CSSCAL	Scale a complex vector.
D1A6	DSCAL	D.P. vector scale $x = a*x$.
D1A6	SSCAL	S.P. vector scale $x = a*x$.
D1A7	CAXPY	Complex computation $y = a*x + y$.
D1A7	DAXPY	D.P computation $y = a*x + y$.
D1A7	SAXPY	S.P. computation $y = a*x + y$.
D1A8	DROT	APPLY d.p. Givens rotation.
D1A8	DROTM	Apply d.p. modified Givens transformation.
D1A8	SROT	Apply s.p. Givens rotation.
D1A8	SROTM	Apply s.p. modified Givens transformation.
D1B10	CROTG	Construct a complex Givens transformation.
D1B10	CSROT	Applies a plane rotation to complex vectors.

D1B10	DROTG	Construct d.p. plane Givens rotation.
D1B10	DROTMG	Construct d.p. modified Givens transformation.
D1B10	SROTG	Construct s.p. plane Givens rotation.
D1B10	SROTMG	Construct s.p. modified Givens transformation.
D2A1	DGECO	Factors a double precision matrix by Gaussian elimination and estimates the condition of the matrix.
D2A1	DGEFA	Factors a double precision matrix by Gaussian elimination.
D2A1	DGEFS	DGEFS solves a GENERAL double precision NXN system of linear equations.
D2A1	DGESL	Solves the double precision system $A \cdot X = B$ or $TRANS(A) \cdot X = B$ using the factors computed by DGECO or DGEFA.
D2A1	SGECO	Factors a real matrix by Gaussian elimination and estimates the condition number of the matrix.
D2A1	SGEDI	Computes the determinant and inverse of a matrix using the factors computed by SGECO or SGEFA.
D2A1	SGEFA	Factors a real matrix by Gaussian elimination.
D2A1	SGEFS	SGEFS solves a GENERAL single precision real NXN system of linear equations.
D2A1	SGEIR	SGEIR solves a GENERAL single precision real NXN system of linear equations. Iterative refinement is used to obtain an error estimate.
D2A1	SGESL	Solves the real system $A \cdot X = B$ or $TRANS(A) \cdot X = B$ using the factors of SGECO or SGEFA.
D2A2	DGBCO	Factors a double precision BAND matrix by Gaussian elimination and estimates the condition of the matrix.
D2A2	DGBFA	Factors a double precision BAND matrix by elimination.
D2A2	DGBSL	Solves the double precision BAND system $A \cdot X = B$ or $TRANS(A) \cdot X = B$ using the factors computed by DGBCO or DGBFA.

D2A2	DNBCO	Factors a double precision BAND matrix by Gaussian elimination and estimates the condition of the matrix.
D2A2	DNBFA	Factors a double precision BAND matrix by elimination.
D2A2	DNBFS	DNBFS solves a general nonsymmetric BANDED double precision NXN system of linear equations.
D2A2	DNBSL	Solves a double precision BAND system using the factors computed by DNBCO or DNBFA.
D2A2	SGBCO	Factors a real BAND matrix by Gaussian elimination and estimates the condition number of the matrix.
D2A2	SGBFA	Factors a real BAND matrix by elimination.
D2A2	SGBSL	Solves the real BAND system $A \cdot X = B$ or $TRANS(A) \cdot X = B$ using the factors computed by SGBCO or SGBFA.
D2A2	SNBCO	SNBCO factors a real BAND matrix by Gaussian elimination and estimates the condition number.
D2A2	SNBFA	SNBFA factors a real BAND matrix by elimination.
D2A2	SNBFS	SNBFS solves a GENERAL NONSYMMETRIC BANDED NXN system of linear equations.
D2A2	SNBIR	SNBIR solves a GENERAL NONSYMMETRIC BANDED NXN system of linear equations. Iterative refinement is used to obtain an error estimate.
D2A2	SNBSL	SNBSL solves a real BAND system using factors computed by SNBCO or SNBFA.
D2A2A	DGTSL	Solves the system $T \cdot X = B$ where T is a general TRIDIAGONAL matrix.
D2A2A	SGTSL	Solves the system $A \cdot X = B$ where a is TRIDIAGONAL
D2A3	DTRCO	Estimates the condition of a double precision TRIANGULAR matrix.
D2A3	DTRDI	Computes the determinant and inverse of a d.p. TRIANGULAR matrix.
D2A3	DTRSL	Solves systems of the form $T \cdot X = B$ or $TRANS(T) \cdot X = B$ where T is a TRIANGULAR matrix of order N.
D2A3	STRCO	Estimates the condition of a real TRIANGULAR matrix.

D2A3	STRDI	Computes the determinant and inverse of a real TRIANGULAR matrix.
D2A3	STRSL	Solves systems of the form $T*X=B$ or $TRANS(T)*X=B$ where T is a TRIANGULAR matrix of order N.
D2B1A	DSICO	Factors a d.p. SYMMETRIC matrix by elimination with symmetric pivoting and estimates the condition of the matrix.
D2B1A	DSIDI	Computes the determinant, inertia and inverse of a d.p. SYMMETRIC matrix using the factors from DSIFA.
D2B1A	DSIFA	Factors a d.p. SYMMETRIC matrix by elimination with symmetric pivoting.
D2B1A	DSISL	Solves the double precision SYMMETRIC system $A*X=B$ using the factors computed by DSIFA.
D2B1A	DSPCO	Factors a d.p. SYMMETRIC matrix stored in packed form by elimination with symmetric pivoting and estimates the condition of the matrix.
D2B1A	DSPDI	Computes the determinant, inertia and inverse of a double precision SYMMETRIC matrix using the factors from DSPFA, where the matrix is stored in packed form.
D2B1A	DSPFA	Factors a double precision SYMMETRIC matrix stored in packed form by elimination with symmetric pivoting.
D2B1A	DSPSL	Solves the double precision SYMMETRIC system $A*X=B$ using the factors computed by DSPFA.
D2B1A	SSICO	Factors a real SYMMETRIC matrix by elimination with symmetric pivoting and estimates the condition of the matrix.
D2B1A	SSIDI	Computes the determinant, inertia and inverse of a real SYMMETRIC matrix using the factors from SSIFA.
D2B1A	SSIFA	Factors a real SYMMETRIC matrix by elimination with symmetric pivoting.
D2B1A	SSISL	Solves the real SYMMETRIC system $A*X=B$ using the factors of SSIFA.
D2B1A	SSPCO	Factors a real SYMMETRIC matrix stored in packed form by elimination with symmetric pivoting and estimates the condition of the matrix.

D2B1A	SSPDI	Computes the determinant, inertia, inverse of a real SYMMETRIC matrix (packed form) using the factors from SSPFA.
D2B1A	SSPFA	Factors a real SYMMETRIC matrix stored in packed form by elimination with symmetric pivoting.
D2B1A	SSPSL	Solves the real SYMMETRIC system $A \cdot X = B$ using factors from SSPFA.
D2B1B	DCHDC	Computes the Cholesky decomposition of a POSITIVE DEFINITE matrix. A pivoting option allows the user to estimate the condition of a positive definite matrix or determine the rank of a POSITIVE SEMIDEFINITE matrix.
D2B1B	DPOCO	Factors a double precision SYMMETRIC POSITIVE DEFINITE matrix and estimates the condition of the matrix.
D2B1B	DPODI	Computes the determinant and inverse of a certain double precision SYMMETRIC POSITIVE DEFINITE matrix (see abstract) using the factors computed by DPOCO, DPOFA or DQRDC.
D2B1B	DPOFA	Factors a double precision SYMMETRIC POSITIVE DEFINITE matrix.
D2B1B	DPOFS	DPOFS solves a POSITIVE DEFINITE SYMMETRIC double precision $N \times N$ system of linear equations.
D2B1B	DPOSL	Solves the double precision SYMMETRIC POSITIVE DEFINITE system $A \cdot X = B$ using the factors computed by DPOCO or DPOFA.
D2B1B	DPFCO	Factors a double precision SYMMETRIC POSITIVE DEFINITE matrix stored in packed form and estimates the condition of the matrix.
D2B1B	DPPDI	Computes the determinant and inverse of a d.p. SYMMETRIC POSITIVE DEFINITE matrix using factors from DPFCO or DPPFA.
D2B1B	DPPFA	Factors a d.p. SYMMETRIC POSITIVE DEFINITE matrix (packed form).
D2B1B	DPPSL	Solves the double precision SYMMETRIC POSITIVE DEFINITE system $A \cdot X = B$.

D2B1B	SCHDC	Computes the Cholesky decomposition of A POSITIVE DEFINITE matrix. A pivoting option allows the user to estimate the condition of a POSITIVE DEFINITE matrix or determine the rank of a POSITIVE SEMIDEFINITE matrix.
D2B1B	SPOCO	Factors a real SYMMETRIC POSITIVE DEFINITE MATRIX and estimates the condition number of the matrix.
D2B1B	SPODI	Computes the determinant and inverse of a certain REAL SYMMETRIC POSITIVE DEFINITE matrix (see abstract) using the factors computed by SPOCO, SPOFA or SQRDC.
D2B1B	SPOFA	Factors a real SYMMETRIC POSITIVE DEFINITE matrix.
D2B1B	SPOFS	SPOFS solves a POSITIVE DEFINITE SYMMETRIC real NXN system of linear equations.
D2B1B	SPOIR	SPOIR solves a POSITIVE DEFINITE SYMMETRIC real NXN system of linear equations. Iterative refinement is used to obtain an error estimate.
D2B1B	SPOSL	Solves the real SYMMETRIC POSITIVE DEFINITE system $A \cdot X = B$ using the factors computed by SPOCO or SPOFA.
D2B1B	SPPCO	Factors a real SYMMETRIC POSITIVE DEFINITE matrix stored in packed form and estimates the condition of the matrix.
D2B1B	SPPDI	Computes the determinant and inverse of a real SYMMETRIC POSITIVE DEFINITE matrix using factors from SPPCO or SPPFA.
D2B1B	SPPFA	SPPFA factors a real SYMMETRIC POSITIVE DEFINITE matrix (packed form).
D2B1B	SPPSL	Solves the real SYMMETRIC POSITIVE DEFINITE system $A \cdot X = B$ using the factors computed by SPPCO or SPPFA.
D2B2	DPBCO	Factors a d.p. SYMMETRIC POSITIVE DEFINITE matrix stored in band form and estimates the condition of the matrix.
D2B2	DPBFA	Factors a d.p. SYMMETRIC POSITIVE DEFINITE matrix stored in band form.
D2B2	DPBSL	Solves the d.p. SYMMETRIC POSITIVE DEFINITE BAND system $A \cdot X = B$ using the factors computed by DPBCO or DPBFA.
D2B2	SPBCO	Factors a real SYMMETRIC POSITIVE DEFINITE matrix (band form) and estimates the condition of the matrix.

D2B2	SPBFA	Factors a real SYMMETRIC POSITIVE DEFINITE MATRIX (BAND FORM) .
D2B2	SPBSL	Solves the real SYMMETRIC POSITIVE DEFINITE BAND system $A*X=B$ using the factors computed by SPBCO or SPBFA.
D2B2A	DPTSL	Solves the system $T*X=B$ where T is POSITIVE DEFINITE SYMMETRIC TRIDIAGONAL.
D2B2A	SPTSL	Solves the system $A*X=B$ where A is POSITIVE DEFINITE and TRIDIAGONAL.
D2C1	CGECO	Factors a COMPLEX matrix by Gaussian elimination and estimates the condition of the matrix.
D2C1	CGEDI	Computes the determinant and inverse of a COMPLEX matrix using the factors computed by CGECO or CGEFA.
D2C1	CGEFA	Factors a COMPLEX matrix by Gaussian elimination.
D2C1	CGEFS	CGEFS solves a GENERAL COMPLEX NXN system of linear equations.
D2C1	CGEIR	CGEIR solves a GENERAL COMPLEX NXN system of linear equations. Iterative refinement is used to obtain an error estimate.
D2C1	CGESL	Solves the COMPLEX system $A*X=B$ or $CTRANS(A)*X=B$ using the factors computed by CGECO or CGEFA.
D2C2	CGBCO	Factors a COMPLEX band matrix by Gaussian elimination and estimates the condition of the matrix.
D2C2	CGBFA	Factors a COMPLEX band matrix by elimination.
D2C2	CGBSL	Solves the COMPLEX band system $A*X=B$ or $CTRANS(A)*X=B$ using the factors computed by CGBCO or CGBFA.
D2C2	CNBCO	Factors a COMPLEX BAND matrix by Gaussian elimination and estimates the condition of the matrix.
D2C2	CNBFA	Factors a COMPLEX BAND matrix by elimination.
D2C2	CNBFS	CNBFS solves a GENERAL NONSYMMETRIC BANDED COMPLEX NXN system of linear equations.
D2C2	CNBIR	CNBIR solves a GENERAL NONSYMMETRIC BANDED COMPLEX NXN system of linear equations. Iterative refinement is used to obtain an error estimate.

D2C2	CNBSL	Solves a COMPLEX BAND system using factors computed by CNBCO or CNBFA.
D2C2A	CGTSL	Solves a GENERAL COMPLEX TRIDIAGONAL system of equations.
D2C3	CTRCO	Estimates the condition of a COMPLEX TRIANGULAR matrix.
D2C3	CTRDI	Computes the determinant and inverse of a COMPLEX TRIANGULAR matrix.
D2C3	CTRSL	Solves systems of the form $T*X=B$ or $CTRANS(T)*X=B$ where T is a triangular matrix. Here CTRANS(T) is conjugate transpose.
D2D1A	CHICO	Factors a COMPLEX HERMITIAN matrix by elimination with symmetric pivoting and estimates the condition of the matrix.
D2D1A	CHIDI	Computes the determinant, inertia and inverse of a COMPLEX HERMITIAN matrix using the factors from CHIFA.
D2D1A	CHIFA	Factors a COMPLEX HERMITIAN matrix by elimination (symmetric pivoting).
D2D1A	CHISL	CHISL solves the COMPLEX HERMITIAN system $A*X=B$ using factors of CHIFA.
D2D1A	CHPCO	Factors a COMPLEX HERMITIAN matrix (packed form) by elimination with symmetric pivoting and estimates the condition of the matrix.
D2D1A	CHPDI	Computes the determinant, inertia and inverse of a COMPLEX HERMITIAN matrix (packed form) using the factors from CHPFA.
D2D1A	CHPFA	Factors a COMPLEX HERMITIAN matrix (packed form) by elimination with symmetric pivoting.
D2D1A	CHPSL	Solves the COMPLEX HERMITIAN system $A*X=B$ using factors of CHPFA.
D2D1A	CSICO	Factors a COMPLEX SYMMETRIC matrix by elimination with symmetric pivoting and estimates the condition of the matrix.
D2D1A	CSIDI	Computes the determinant and inverse of a COMPLEX SYMMETRIC matrix using the factors from CSIFA.

D2D1A	CSIFA	Factors a COMPLEX SYMMETRIC MATRIX by elimination (symmetric pivoting).
D2D1A	CSISL	Solves the COMPLEX SYMMETRIC SYSTEM $A^*X=B$ using factors from CSIFA.
D2D1A	CSPCO	Factors a COMPLEX SYMMETRIC matrix stored in packed form by elimination with symmetric pivoting and estimates the condition of the matrix.
D2D1A	CSPDI	Computes the determinant and inverse of a complex symmetric matrix stored in packed form using factors from CSPFA.
D2D1A	CSPFA	Factors a COMPLEX SYMMETRIC matrix stored in packed form by elimination with symmetric pivoting.
D2D1A	CSPSL	Solves the COMPLEX SYMMETRIC system $A^*X=B$ using factors from CSPFA.
D2D1B	CCHDC	Computes the Cholesky decomposition of a positive definite matrix. A pivoting option allows the user to estimate the condition of a positive definite matrix or determine the rank of a positive definite matrix.
D2D1B	CPOCO	Factors a COMPLEX HERMITIAN POSITIVE DEFINITE matrix and estimates the condition of the matrix.
D2D1B	CPODI	Computes the determinant and inverse of a certain COMPLEX HERMITIAN POSITIVE DEFINITE matrix using factors of CPOCO, CPOFA, or CQRDC.
D2D1B	CPOFA	Factors a COMPLEX HERMITIAN POSITIVE DEFINITE matrix.
D2D1B	CPOFS	CPOFS solves a POSITIVE DEFINITE SYMMETRIC COMPLEX NXN system of linear equations.
D2D1B	CPOIR	CPOIR solves a POSITIVE DEFINITE HERMITIAN COMPLEX NXN system of linear equations. Iterative refinement is used to obtain an error estimate.
D2D1B	CPOSL	Solves the COMPLEX HERMITIAN POSITIVE DEFINITE system $A^*X=B$ using the factors computed by CPOCO or CPOFA.
D2D1B	CPPCO	Factors a COMPLEX HERMITIAN POSITIVE DEFINITE matrix stored in packed form and estimates the condition of the matrix.

D2D1B	CPPDI	Computes the determinant and inverse of a COMPLEX HERMITIAN POSITIVE DEFINITE matrix using factors from CPPCO or CPPFA.
D2D1B	CPPFA	Factors a COMPLEX HERMITIAN POSITIVE DEFINITE matrix (packed form).
D2D1B	CPPSL	Solves the COMPLEX HERMITIAN POSITIVE DEFINITE system $A \cdot X = B$ using the factors computed by CPPCO or CPPFA.
D2D2	CPBCO	Factors a COMPLEX HERMITIAN POSITIVE DEFINITE matrix stored in band form and estimates the condition of the matrix.
D2D2	CPBFA	Factors a COMPLEX HERMITIAN POSITIVE DEFINITE matrix (band form).
D2D2	CPBSL	Solves the COMPLEX HERMITIAN POSITIVE DEFINITE BAND system $A \cdot X = B$ using factors from CPBCO or CPBFA.
D2D2A	CPTSL	Solves a COMPLEX POSITIVE DEFINITE TRIDIAGONAL system of equations.
D3A1	DGEDI	Computes the determinant and inverse of a matrix using factors computed by DGECO or DGEFA.
D3A2	DGBDI	Computes the determinant of a BAND matrix using factors computed by DGBCO or DGBFA. Use DGBSL N times for the inverse.
D3A2	DNBDI	Computes the determinant of a Band matrix using the factors computed by DNBCO or DNBFA.
D3A2	SGBDI	Computes the determinant of a BAND matrix using the factors computed by SGBCO or SGBFA. If the inverse is needed, use SGBSL N times.
D3A2	SNBDI	Computes the determinant of a BAND matrix using the factors computed by SNBCO or SNBFA.
D3B2	DPBDI	Computes the determinant of a d.p. SYMMETRIC POSITIVE DEFINITE BAND matrix using factors of DPBCO or DPBFA. Use DPBSL N times for inverse.
D3B2	SPBDI	Computes the determinant of a real SYMMETRIC POSITIVE DEFINITE BAND matrix using the factors computed by SPBCO or SPBFA. If the inverse is needed, use SPBSL N times.

D3C2	CGBDI	Computes the determinant of a COMPLEX band matrix using factors from CGBCO or CGBFA. If the inverse is needed, use CGBSL N times.
D3C2	CNBDI	Computes the determinant of a COMPLEX BAND matrix using factors computed by CNBCO or CNBFA.
D3D2	CPBDI	Computes the determinant of a COMPLEX HERMITIAN POSITIVE DEFINITE BAND matrix using factors from CPBCO or CPBFA. Use CPBSL for inverse using the factors computed by CPBCO or CPBFA.
D4	EISDOC	Documentation for EISPACK.
D4A1	RS	Computes eigenvalues and, optionally, eigenvectors of real symmetric matrix.
D4A1	RSP	Compute eigenvalues and, optionally, eigenvectors of real symmetric matrix packed into a one dimensional.
D4A1	SSIEV	To compute the eigenvalues and, optionally, the eigenvectors of a real SYMMETRIC matrix.
D4A1	SSPEV	To compute the eigenvalues and, optionally, the eigenvectors of a real SYMMETRIC matrix stored in packed form.
D4A2	RG	Computes eigenvalues and, optionally, eigenvectors of a real general matrix.
D4A2	SGEEV	To compute the eigenvalues and, optionally, the eigenvectors of a GENERAL real matrix.
D4A3	CH	Computes the eigenvalues and, optionally, eigenvector of a complex Hermitian matrix.
D4A3	CHIEV	To compute the eigenvalues and, optionally, the eigenvectors of a COMPLEX HERMITIAN matrix.
D4A4	CG	Computes the eigenvalues and, optionally, the eigenvectors of a complex general matrix.
D4A4	CGEEV	To compute the eigenvalues and, optionally, the eigenvectors of a GENERAL COMPLEX matrix.
D4A5	BISECT	Compute eigenvalues of symmetric tridiagonal matrix given interval using Sturm sequencing.

D4A5	IMTQL1	Computes eigenvalues of symmetric tridiagonal matrix implicit QL method.
D4A5	IMTQL2	Computes eigenvalues and eigenvectors of symmetric tridiagonal matrix using implicit QL method.
D4A5	IMTQLV	Computes eigenvalues of symmetric tridiagonal matrix by the implicit QL method. Eigenvectors may be computed later.
D4A5	RATQR	Computes largest or smallest eigenvalues of symmetric tridiagonal matrix using rational QR method with Newton correction.
D4A5	RST	Compute eigenvalues and, optionally, eigenvectors of real symmetric tridiagonal matrix.
D4A5	RT	Compute eigenvalues and eigenvectors of a special real tridiagonal matrix.
D4A5	TQL1	Compute eigenvalues of symmetric tridiagonal matrix by QL method.
D4A5	TQL2	Compute eigenvalues and eigenvectors of symmetric tridiagonal matrix.
D4A5	TQLRAT	Computes eigenvalues of symmetric tridiagonal matrix a rational variant of the QL method.
D4A5	TRIDIB	Computes eigenvalues of symmetric tridiagonal matrix given interval using Sturm sequencing.
D4A5	TSTURM	Computes eigenvalues of symmetric tridiagonal matrix given interval and eigenvectors by Sturm sequencing. This subroutine is a translation of the ALGOL procedure TRISTURM by Peters and Wilkinson. HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 418-439(1971).
D4A6	BQR	Computes some of the eigenvalues of a real symmetric matrix using the QR method with shifts of origin.
D4A6	RSB	Computes eigenvalues and, optionally, eigenvectors of symmetric band matrix.
D4B1	RSG	Computes eigenvalues and, optionally, eigenvectors of symmetric generalized eigenproblem: $A*X=(LAMBDA)*B*X$.
D4B1	RSGAB	Computes eigenvalues and, optionally, eigenvectors of symmetric generalized eigenproblem: $A*B*X=(LAMBDA)*X$.

D4B1	RSGBA	Computes eigenvalues and, optionally, eigenvectors of symmetric generalized eigenproblem: $B^*A^*X = (\text{LAMBDA})^*X$.
D4B2	RGG	Computes eigenvalues and eigenvectors for real generalized eigenproblem: $A^*X = (\text{LAMBDA})^*B^*X$.
D4C1A	BALANC	Balances a general real matrix and isolates eigenvalue whenever possible.
D4C1A	CBAL	Balances a complex general matrix and isolates eigenvalues whenever possible.
D4C1B1	BANDR	Reduces real symmetric band matrix to symmetric tridiagonal matrix and, optionally, accumulates orthogonal similarity transformations.
D4C1B1	HTRID3	Reduces complex Hermitian (packed) matrix to real symmetric tridiagonal matrix by unitary similarity transformations.
D4C1B1	HTRIDI	Reduces complex Hermitian matrix to real symmetric tridiagonal matrix using unitary similarity transformations.
D4C1B1	TRED1	Reduce real symmetric matrix to symmetric tridiagonal matrix using orthogonal similarity transformations.
D4C1B1	TRED2	Reduce real symmetric matrix to symmetric tridiagonal matrix using and accumulating orthogonal transformation.
D4C1B1	TRED3	Reduce real symmetric matrix stored in packed form to symmetric tridiagonal matrix using orthogonal transformations.
D4C1B2	COMHES	Reduces complex general matrix to complex upper Hess form using stabilized elementary similarity transformations.
D4C1B2	CORTH	Reduces complex general matrix to complex upper Hessenberg using unitary similarity transformations.
D4C1B2	ELMHES	Reduces real general matrix to upper Hessenberg form stabilized elementary similarity transformations.
D4C1B2	ORTHES	Reduces real general matrix to upper Hessenberg form orthogonal similarity transformations.

D4C1B3	QZHES	The first step of the QZ algorithm for solving generalized matrix eigenproblems. Accepts a pair of real general matrices and reduces one of them to upper Hessenberg and the other to upper triangular form using orthogonal transformations. Usually followed by QZIT, QZVAL, and QZ.
D4C1B3	QZIT	The second step of the QZ algorithm for generalized eigenproblems. Accepts an upper Hessenberg and an upper triangular matrix and reduces the former to quasi-triangular form while preserving the form of the latter. Usually preceded by QZHES and followed by QZVAL and QZVEC.
D4C1C	FIGI	Transforms certain real non-symmetric tridiagonal matrix to symmetric tridiagonal matrix.
D4C1C	FIGI2	Transforms certain real non-symmetric tridiagonal matrix to symmetric tridiagonal matrix.
D4C1C	REDUC	Reduces generalized symmetric eigenproblem $A*X=(LAMBDA)*B*X$, to standard symmetric eigenproblem using Cholesky factorization.
D4C1C	REDUC2	Reduces certain generalized symmetric eigenproblems standard symmetric eigenproblem, using Cholesky factorization.
D4C2B	CINVIT	Computes eigenvectors of a complex upper Hessenberg associated with specified eigenvalues using inverse iteration.
D4C2B	COMLR	Computes eigenvalues of a complex upper Hessenberg matrix using the modified LR method.
D4C2B	COMLR2	Computes eigenvalues and eigenvectors of complex upper Hessenberg matrix using modified LR method.
D4C2B	COMQR	Computes eigenvalues of complex upper Hessenberg matrix using the QR method.
D4C2B	COMQR2	Computes eigenvalues and eigenvectors of complex upper Hessenberg matrix.
D4C2B	HQR	Computes eigenvalues of a real upper Hessenberg matrix using the QR method.
D4C2B	HQR2	Computes eigenvalues and eigenvectors of real upper Hessenberg matrix using QR method.

D4C2B	INVIT	Computes eigenvectors of upper Hessenberg (real) matrix associated with specified eigenvalues by inverse iteration.
D4C2C	QZVAL	The third step of the QZ algorithm for generalized eigenproblems. Accepts a pair of real matrices, one quasi-triangular form and the other in upper triangular form and computes the eigenvalues of the associated eigenproblem. Usually preceded by QZHES, QZIT, and followed by QZVEC.
D4C3	BANDV	Forms eigenvectors of real symmetric band matrix associated with a set of ordered approximate eigenvalue by inverse iteration.
D4C3	QZVEC	The optional fourth step of the QZ algorithm for generalized eigenproblems. Accepts a matrix in quasi-triangular form and another in upper triangular and computes the eigenvectors of the triangular problem and transforms them back to the original coordinates. Ususally preceded by QZHES, QZIT, QZVAL.
D4C3	TINVIT	Eigenvectors of symmetric tridiagonal matrix corresponding to some specified eigenvalues, using inverse iteration.
D4C4	BAKVEC	Forms eigenvectors of certain real non-symmetric tridiagonal matrix from symmetric tridiagonal matrix output from FIGI.
D4C4	BALBAK	Forms eigenvectors of real general matrix from eigenvectors of matrix output from BALANC.
D4C4	CBABK2	Forms eigenvectors of complex general matrix from eigenvectors of matrix output from CBAL.
D4C4	COMBAK	Forms eigenvectors of complex general matrix from eigenvectors of upper Hessenberg matrix output from COMHES.
D4C4	CORTB	Forms eigenvectors of complex general matrix from eigenvectors of upper Hessenberg matrix output from CORTH.
D4C4	ELMBAK	Forms eigenvectors of real general matrix from eigenvectors of upper Hessenberg matrix output from ELMHES.

D4C4	ELTRAN	Accumulates the stabilized elementary similarity transformations used in the reduction of a real general matrix to upper Hessenberg form by ELMHES.
D4C4	HTRIB3	Computes eigenvectors of complex Hermitian matrix from eigenvectors of real symmetric tridiagonal matrix output from HTRID3.
D4C4	HTRIBK	Forms eigenvectors of complex Hermitian matrix from eigenvectors of real symmetric tridiagonal matrix output from HTRIDI.
D4C4	ORTBAK	Forms eigenvectors of general real matrix from eigenvectors of upper Hessenberg matrix output from ORTHES.
D4C4	ORTRAN	Accumulates orthogonal similarity transformations in reduction of real general matrix by ORTHES.
D4C4	REBAK	Forms eigenvectors of generalized symmetric eigensystem from eigenvectors of derived matrix output from REDUC2.
D4C4	REBAKB	Forms eigenvectors of generalized symmetric eigensystem from eigenvectors of derived matrix output from REDUC2.
D4C4	TRBAK1	Forms the eigenvectors of real symmetric matrix from eigenvectors of symmetric tridiagonal matrix formed TRED1.
D4C4	TRBAK3	Forms eigenvectors of real symmetric matrix from the eigenvectors of symmetric tridiagonal matrix formed TRED3.
D5	CQRDC	Uses Householder transformations to compute the QR factorization of an N by P matrix X. Column pivoting is optional.
D5	DQRDC	Uses Householder transformations to compute the Qr factorization of N by P matrix X. Column pivoting is optional.
D5	SQRDC	Uses Householder transformations to compute the QR factorization of an N by P matrix X. Column pivoting is a users option.
D6	CSVDC	Perform the singular value decomposition of a COMPLEX NXP matrix.

D6	DSVDC	Perform the singular value decomposition of a d.p. NXP matrix.
D6	SSVDC	Perform the singular value decomposition of a real NXP matrix.
D7B	CCHDD	Downdates an augmented Cholesky decomposition or the triangular factor of an augmented QR decomposition.
D7B	CCHEX	CCHEX updates the Cholesky factorization $A = CTRANS(R) * R$ of a positive definite matrix A of order P under diagonal permutations of the form $U * R * E = RR$, where E is a permutation matrix.
D7B	CCHUD	Updates an augmented Cholesky decomposition of the triangular part of an augmented QR decomposition.
D7B	DCHDD	Downdates an augmented Cholesky decomposition or the triangular factor of an augmented QR decomposition.
D7B	DCHEX	Updates the Cholesky factorization $A = TRANS(R) * R$ of a POSITIVE DEFINITE matrix A of order P under diagonal permutations of the form $TRANS(E) * A * E$ where E is a permutation matrix.
D7B	DCHUD	Updates an augmented Cholesky decomposition of the triangular part of an augmented QR decomposition.
D7B	SCHDD	Downdates an augmented Cholesky decomposition or the triangular factor of an augmented QR decomposition.
D7B	SCHEX	Updates the Cholesky factorization $A = TRANS(R) * R$ of A POSITIVE DEFINITE matrix A of order P under diagonal permutations of the form $TRANS(E) * A * E$ where E is a permutation matrix.
D7B	SCHUD	Updates an augmented Cholesky decomposition of the triangular part of an augmented QR decomposition.
D9	BNDACC	Solve the least squares problem $Ax = b$ for banded matrices A using sequential accumulation of rows of the data matrix. Exactly one right-handed side vector is permitted.
D9	BNDSOL	Solve the least squares problem $Ax = b$ for banded matrices A using sequential accumulation of rows of the data matrix. Exactly one right-handed side vector is permitted.

D9	CQRSL	Applies the output of CQRDC to compute coordinate transformations, projections, and least squares solutions.
D9	DQRSL	Applies the output of DQRDC to compute coordinate transformations, projections, and least squares solutions.
D9	HFTI	Solve the least squares problem $Ax = b$ for banded matrices A using sequential accumulation of rows of the data matrix. Exactly one right-handed side vector is permitted.
D9	LLSIA	Solves LINEAR LEAST SQUARES problems by performing a QR factorization of the matrix A using Householder transformations. Emphasis is put on detecting possible rank deficiency.
D9	MINFIT	Compute Singular Value Decomposition of rectangular matrix and solve related Linear Least Squares problem.
D9	SGLSS	Solves LINEAR LEAST SQUARES problems by performing a QR factorization of the matrix A using Householder transformations. Emphasis is put on detecting possible rank deficiency.
D9	SQRSL	Applies the output of SQRDC to compute coordinate transformations projections, and least squares solutions.
D9	ULSIA	Solves the UNDERDETERMINED LINEAR system of equations by performing an LQ factorization of the matrix A using Householder transformations. Emphasis is put on detecting possible rank deficiency.
E	BSPDOC	Comments on B-spline routines.
E1A	BINT4	Computes the B representation of a cubic spline which interpolates data $(X(I), Y(I)), I=1, NDATA$.
E1A	BINTK	Produces the B-spline coefficients, BCOEF, of the B-spline of order K with knots $T(I), I=1, \dots, N+K$, which takes on the value $Y(I)$ at $X(I), I=1, \dots, N$.
E1A	DBINT4	Computes the B-representation of a cubic spline which interpolates data $(X(I), Y(I)), I=1, NDATA$.
E1A	DBINTK	Produces the B-spline coefficients, BCOEF, of the B-spline of order K with knots $T(I), I=1, \dots, N+K$, which takes on the value $Y(I)$ at $X(I), I=1, \dots, N$.

E1B	POLINT	To produce the polynomial which interpolates a set of discrete data points.
E3	BSPDR	Uses the B-representation to construct a divided difference table preparatory to a (right) derivative calculation in BSPEV.
E3	BSPEV	Calculates the value of the spline and its derivatives at X from the B-representation.
E3	BSPPP	Converts the B-representation to the piecewise polynomial (PP) form for use with PPVAL.
E3	BSPVD	Calculates the value and all derivatives of order less than NDERIV of all basis functions which do not vanish at X.
E3	BSPVN	Calculates the value of all (possibly) nonzero basis functions at X.
E3	BVALU	Evaluates the B-representation of a B-spline at X for the function value or any of its derivatives.
E3	DBSPDR	Uses the B-representation to construct a divided difference table ad preparatory to a (right) derivative calculation in BSPEV.
E3	DBSPEV	Calculates the value of the spline and its derivatives at X from the B-representation.
E3	DBSPPP	Converts the B-representation to the piecewise polynomial (PP) form for use with PPVAL.
E3	DBSPVD	Calculates the value and all derivatives of order less than NDERIV of all basis functions which do not vanish at X.
E3	DBSPVN	Calculates the value of all (possibly) nonzero basis functions at X.
E3	DBVALU	Evaluates the B-representation of a B-spline at X for the function value or any of its derivatives.
E3	DINTRV	Computes the largest integer ILEFT in 1.LE.ILEFT.LE.LXT such that XT(ILEFT).LE.X where XT(*) is a subdivision of the X interval.
E3	DPPVAL	Calculates (at X) the value of the IDERIV-th derivative of the B-spline from the PP-representation.

E3	INTRV	Computes the largest integer ILEFT in 1.LE.ILEFT.LE.LXT such that $XT(ILEFT).LE.X$ where $XT(*)$ is a subdivision of the X interval.
E3	POLYVL	Calculates the value of the polynomial and its first n derivatives where the polynomial was produced by a previous call to POLINT.
E3	PPVAL	Calculates (at X) the value of the IDERIV-th derivative of the B-spline from the PP-representation.
F1A1A	RPQR79	To find the zeros of a polynomial with real coefficients.
F1A1A	RPZERO	Find the zeros of a polynomial with real coefficients.
F1A1B	CPQR79	Find the zeros of a polynomial with complex coefficients.
F1A1B	CPZERO	Find the zeros of a polynomial with complex coefficients.
F1B	FZERO	FZERO searches for a zero of a function F(X) in a given interval (B,C). It is designed primarily for problems where F(B) and F(C) have opposite signs.
F2A	DNSQ	To find a zero of a system of N nonlinear functions in N variables by a modification of the Powell hybrid method. This code is the combination of the MINPACK codes (Argonne) HYBRD and HYBRDJ.
F2A	DNSQE	The easy-to-use version of DNSQ which finds a zero of a system of N nonlinear functions in N variables by a modification of the Powell hybrid method. This code is a combination of the MINPACK codes HYBRD1 and HYBRJ1.
F2A	DSOS	DSOS solves a square system of nonlinear equations.
F2A	SNSQ	SNSQ finds to find a zero of a system of N nonlinear functions in N variables by a modification of the Powell hybrid method. This code is the combination of the MINPACK codes (Argonne) HYBRD and HYBRDJ.
F2A	SNSQE	SNSQE is the easy-to-use version of SNSQ which finds a zero of a system of N nonlinear functions in N variables by a modification of Powell hybrid method. This code is the combination of the MINPACK codes Argonne) HYBRD1 and HYBRJ1.
F2A	SOS	SOS Solves a square system of nonlinear equations.

F3	CHKDER	Checks the gradients of M nonlinear functions in N variables, evaluated at a point X, for consistency with the functions themselves.
F3	DCKDER	Checks the gradients of M nonlinear functions in N variables, evaluated at a point X, for consistency with the functions themselves.
H2	QPDOG	QUADPACK documentation routine.
H2A1A1	DGAUS8	DGAUS8 integrates real functions of one variable over finite intervals using an adaptive 8-point Legendre-Gauss algorithm. DGAUS8 is intended primarily for high accuracy integration or integration of smooth functions.
H2A1A1	DQAG	The routine calculates an approximation result to a given definite integral $I = \text{integral of } F \text{ over } (A,B)$, hopefully satisfying following claim for accuracy $\text{ABS}(I - \text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.
H2A1A1	DQAGE	The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F \text{ over } (A,B)$, hopefully satisfying following claim for accuracy $\text{ABS}(I - \text{RESLT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.
H2A1A1	DQAGS	The routine calculates an approximation result to a given Definite integral $I = \text{Integral of } F \text{ over } (A,B)$, Hopefully satisfying following claim for accuracy $\text{ABS}(I - \text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.
H2A1A1	DQAGSE	The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F \text{ over } (A,B)$, hopefully satisfying following claim for accuracy $\text{ABS}(I - \text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.
H2A1A1	DQNC79	Integrate a user defined function by a 7-point adaptive Newton-Cotes quadrature rule.
H2A1A1	DQNG	The routine calculates an approximation result to a given definite integral $I = \text{integral of } F \text{ over } (A,B)$, hopefully satisfying following claim for accuracy $\text{ABS}(I - \text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.
H2A1A1	GAUS8	GAUS8 integrates real functions of one variable over finite intervals using an adaptive 8-point Legendre-Gauss algorithm. GAUS8 is intended primarily for high accuracy integration or integration of smooth functions.

H2A1A1	QAG	The routine calculates an approximation result to a given definite integral $I = \text{integral of } F \text{ over } (A,B)$, hopefully satisfying following claim for accuracy $\text{ABS}(I - \text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.
H2A1A1	QAGE	The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F \text{ over } (A,B)$, hopefully satisfying following claim for accuracy $\text{ABS}(I - \text{RESLT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.
H2A1A1	QAGS	The routine calculates an approximation result to a given Definite integral $I = \text{Integral of } F \text{ over } (A,B)$, Hopefully satisfying following claim for accuracy $\text{ABS}(I - \text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.
H2A1A1	QAGSE	The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F \text{ over } (A,B)$, hopefully satisfying following claim for accuracy $\text{ABS}(I - \text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.
H2A1A1	QNC79	Integrate a user defined function by a 7-point adaptive Newton-Cotes quadrature rule.
H2A1A1	QNG	The routine calculates an approximation result to a given definite integral $I = \text{integral of } F \text{ over } (A,B)$, hopefully satisfying following claim for accuracy $\text{ABS}(I - \text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.
H2A1A2	DQK15	To compute $I = \text{Integral of } F \text{ over } (A,B)$, with error estimate $J = \text{integral of } \text{ABS}(F) \text{ over } (A,B)$
H2A1A2	DQK21	To compute $I = \text{Integral of } F \text{ over } (A,B)$, with error estimate $J = \text{Integral of } \text{ABS}(F) \text{ over } (A,B)$
H2A1A2	DQK31	To compute $I = \text{Integral of } F \text{ over } (A,B)$ with error estimate $J = \text{Integral of } \text{ABS}(F) \text{ over } (A,B)$
H2A1A2	DQK41	To compute $I = \text{Integral of } F \text{ over } (A,B)$, with error estimate $J = \text{Integral of } \text{ABS}(F) \text{ over } (A,B)$
H2A1A2	DQK51	To compute $I = \text{Integral of } F \text{ over } (A,B)$ with error estimate $J = \text{Integral of } \text{ABS}(F) \text{ over } (A,B)$

H2A1A2	DQK61	To compute $I = \text{Integral of } F \text{ over } (A,B) \text{ with error estimate}$ $J = \text{Integral of DABS}(F) \text{ over } (A,B)$
H2A1A2	QK15	To compute $I = \text{Integral of } F \text{ over } (A,B), \text{ with error estimate}$ $J = \text{integral of ABS}(F) \text{ over } (A,B)$
H2A1A2	QK21	To compute $I = \text{Integral of } F \text{ over } (A,B), \text{ with error estimate}$ $J = \text{Integral of ABS}(F) \text{ over } (A,B)$
H2A1A2	QK31	To compute $I = \text{Integral of } F \text{ over } (A,B) \text{ with error estimate}$ $J = \text{Integral of ABS}(F) \text{ over } (A,B)$
H2A1A2	QK41	To compute $I = \text{Integral of } F \text{ over } (A,B), \text{ with error estimate}$ $J = \text{Integral of ABS}(F) \text{ over } (A,B)$
H2A1A2	QK51	To compute $I = \text{Integral of } F \text{ over } (A,B) \text{ with error estimate}$ $J = \text{Integral of ABS}(F) \text{ over } (A,B)$
H2A1A2	QK61	To compute $I = \text{Integral of } F \text{ over } (A,B) \text{ with error estimate}$ $J = \text{Integral of DABS}(F) \text{ over } (A,B)$
H2A1B2	AVINT	Integrate a function tabulated at arbitrarily spaced abscissas using overlapping parabolas.
H2A1B2	DAVINT	Integrate a function tabulated at arbitrarily spaced abscissas using overlapping parabolas.
H2A2A1	BFQAD	Computes the integral on $(X1,X2)$ of a product of a function F and the ID -th derivative of a K -th order B-spline (B-representation).
H2A2A1	BSQAD	Computes the integral on $(X1,X2)$ of a K -th order B-spline using the B-representation.
H2A2A1	DBFQAD	Computes the integral on $(X1,X2)$ of a product of a function F and the ID -th derivative of a K -th order B-spline (B-representation).
H2A2A1	DBSQAD	Computes the integral on $(X1,X2)$ of a K -th order B-spline using the B-representation.

H2A2A1	DPFQAD	Computes the integral on (X1,X2) of a product of a function F and the ID-th derivative of a B-spline, (PP-representation).
H2A2A1	DPPQAD	Computes the integral on (X1,X2) of a K-th order B-spline using the piecewise polynomial representation.
H2A2A1	DQAGP	The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F \text{ over } (A,B)$, hopefully satisfying following claim for accuracy break points of the integration interval, where local difficulties of the integrand may occur (e.g. SINGULARITIES, DISCONTINUITIES), are provided by the user.
H2A2A1	DQAGPE	The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F \text{ over } (A,B)$, hopefully satisfying following claim for accuracy $\text{ABS}(I-\text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$. Break points of the integration interval, where local difficulties of the integrand may occur (e.g. singularities, discontinuities), provided by user.
H2A2A1	DQAWC	The routine calculates an approximation result to a Cauchy principal value $I = \text{INTEGRAL of } F * W \text{ over } (A,B)$ ($W(X) = 1 / (X - C)$, $C.NE.A$, $C.NE.B$), hopefully satisfying following claim for accuracy $\text{ABS}(I-\text{RESULT}) \leq \text{MAX}(\text{EPSABE}, \text{EPSREL} * \text{ABS}(I))$.
H2A2A1	DQAWCE	The routine calculates an approximation result to a CAUCHY PRINCIPAL VALUE $I = \text{Integral of } F * W \text{ over } (A,B)$ ($W(X) = 1 / (X - C)$, $(C.NE.A, C.NE.B)$, hopefully satisfying following claim for accuracy $\text{ABS}(I-\text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.
H2A2A1	DQAWO	The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F(X) * W(X) \text{ over } (A,B)$ WHERE $W(X) = \text{COS}(\text{OMEGA} * X)$ OR $W(X) = \text{SIN}(\text{OMEGA} * X)$, hopefully satisfying following claim for accuracy $\text{ABS}(I-\text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.
H2A2A1	DQAWOE	The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F(X) * W(X) \text{ over } (A,B)$ where $W(X) = \text{COS}(\text{OMEGA} * X)$ or $W(X) = \text{SIN}(\text{OMEGA} * X)$, hopefully satisfying following claim for accuracy $\text{ABS}(I-\text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.

H2A2A1	DQAWS	The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F \cdot W \text{ over } (A,B)$, (where W shows a singular behaviour at the end points see parameter INTEGR). Hopefully satisfying following claim for accuracy $\text{ABS}(I-\text{RESULT})$.LE. $\text{MAX}(\text{EPSABS}, \text{EPSREL} \cdot \text{ABS}(I))$.
H2A2A1	DQAWSE	The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F \cdot W \text{ over } (A,B)$, (where W shows a singular behaviour at the end points, see parameter INTEGR). Hopefully satisfying following claim for accuracy $\text{ABS}(I-\text{RESULT})$.LE. $\text{MAX}(\text{EPSABS}, \text{EPSREL} \cdot \text{ABS}(I))$.
H2A2A1	DQMOMO	This routine computes modified CHEBSYSHEV moments. The K -th modified CHEBYSHEV moment is defined as the integral over $(-1,1)$ of $W(X) \cdot T(K,X)$, where $T(K,X)$ is the CHEBYSHEV POLYNOMIAL of degree K .
H2A2A1	PFQAD	Computes the integral on $(X1,X2)$ of a product of a function F and the ID -th derivative of a B-spline, (PP-representation).
H2A2A1	PPQAD	Computes the integral on $(X1,X2)$ of a K -th order B-spline using the piecewise polynomial representation.
H2A2A1	QAGP	The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F \text{ over } (A,B)$, hopefully satisfying following claim for accuracy break points of the integration interval, where local difficulties of the integrand may occur (e.g. SINGULARITIES, DISCONTINUITIES), are provided by the user.
H2A2A1	QAGPE	The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F \text{ over } (A,B)$, hopefully satisfying following claim for accuracy $\text{ABS}(I-\text{RESULT})$.LE. $\text{MAX}(\text{EPSABS}, \text{EPSREL} \cdot \text{ABS}(I))$. Break points of the integration interval, where local difficulties of the integrand may occur (e.g. singularities, discontinuities) provided by user.
H2A2A1	QAWC	The routine calculates an approximation result to a Cauchy principal value $I = \text{INTEGRAL of } F \cdot W \text{ over } (A,B)$ ($W(X) = 1/(X-C)$, $C.NE.A$, $C.NE.B$), hopefully satisfying following claim for accuracy $\text{ABS}(I-\text{RESULT})$.LE. $\text{MAX}(\text{EPSABE}, \text{EPSREL} \cdot \text{ABS}(I))$.

H2A2A1	QAWCE	The routine calculates an approximation result to a CAUCHY PRINCIPAL VALUE $I = \text{Integral of } F \cdot W \text{ over } (A,B)$ ($W(X) = 1/(X-C)$, $(C.NE.A, C.NE.B)$, hopefully satisfying following claim for accuracy $ABS(I-RESULT) \leq MAX(EPSABS, EPSREL \cdot ABS(I))$).
H2A2A1	QAWO	The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F(X) \cdot W(X) \text{ over } (A,B)$ where $W(X) = \cos(\Omega \cdot X)$ or $W(X) = \sin(\Omega \cdot X)$, hopefully satisfying following claim for accuracy $ABS(I-RESULT) \leq MAX(EPSABS, EPSREL \cdot ABS(I))$.
H2A2A1	QAWOE	The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F(X) \cdot W(X) \text{ over } (A,B)$ where $W(X) = \cos(\Omega \cdot X)$ or $W(X) = \sin(\Omega \cdot X)$, hopefully satisfying following claim for accuracy $ABS(I-RESULT) \leq MAX(EPSABS, EPSREL \cdot ABS(I))$.
H2A2A1	QAWS	The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F \cdot W \text{ over } (A,B)$, (where W shows a singular behaviour at the end points see parameter INTEGR). Hopefully satisfying following claim for accuracy $ABS(I-RESULT) \leq MAX(EPSABS, EPSREL \cdot ABS(I))$.
H2A2A1	QAWSE	The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F \cdot W \text{ over } (A,B)$, (where W shows a singular behaviour at the end points, see parameter INTEGR). Hopefully satisfying following claim for accuracy $ABS(I-RESULT) \leq MAX(EPSABS, EPSREL \cdot ABS(I))$.
H2A2A1	QMOMO	This routine computes modified CHEBSYSHEV moments. The K -th modified CHEBYSHEV moment is defined as the integral over $(-1,1)$ of $W(X) \cdot T(K,X)$, where $T(K,X)$ is the CHEBYSHEV POLYNOMIAL of degree K .
H2A2A2	DQC25C	To compute $I = \text{Integral of } F \cdot W \text{ over } (A,B)$ with error estimate, where $W(X) = 1/(X-C)$.
H2A2A2	DQC25F	To compute the integral $I = \text{Integral of } F(X) \text{ over } (A,B)$ Where $W(X) = \cos(\Omega \cdot X)$ or $W(X) = \sin(\Omega \cdot X)$ and to compute $J = \text{Integral of } ABS(F) \text{ over } (A,B)$. For small value of Ω or small intervals (A,B) the 15-point GAUSS-KRONRO Rule is used. Otherwise a generalized CLENSHAW-CURTIS method is used.

H2A2A2	DQC25S	To compute $I = \text{Integral of } F*W \text{ over } (BL, BR)$, with error estimate, where the weight function W has a singular behaviour of ALGEBRAICO-LOGARITHMIC type at the points A and/or B . (BL, BR) is a part of (A, B) .
H2A2A2	DQK15W	To compute $I = \text{Integral of } F*W \text{ over } (A, B)$, with error estimate $J = \text{Integral of } ABS(F*W) \text{ over } (A, B)$
H2A2A2	QC25C	To compute $I = \text{Integral of } F*W \text{ over } (A, B)$ with error estimate, where $W(X) = 1/(X-C)$.
H2A2A2	QC25F	To compute the integral $I = \text{Integral of } F(X) \text{ over } (A, B)$ Where $W(X) = \cos(\text{OMEGA}*X)$ Or $(WX) = \sin(\text{OMEGA}*X)$ and to compute $J = \text{Integral of } ABS(F) \text{ over } (A, B)$. For small value of OMEGA or small intervals (A, B) 15-point GAUSS-KRONROD Rule used. Otherwise generalized CLENSHAW-CURTIS used.
H2A2A2	QC25S	To compute $I = \text{Integral of } F*W \text{ over } (BL, BR)$, with error estimate, where the weight function W has a singular behaviour of ALGEBRAICO-LOGARITHMIC type at the points A and/or B . (BL, BR) is a part of (A, B) .
H2A2A2	QK15W	To compute $I = \text{Integral of } F*W \text{ over } (A, B)$, with error estimate $J = \text{Integral of } ABS(F*W) \text{ over } (A, B)$
H2A3A1	DQAGI	The routine calculates an approximation result to a given INTEGRAL $I = \text{Integral of } F \text{ over } (\text{BOUND}, +\text{INFINITY})$ OR $I = \text{Integral of } F \text{ over } (-\text{INFINITY}, \text{BOUND})$ OR $I = \text{Integral of } F \text{ over } (-\text{INFINITY}, +\text{INFINITY})$ Hopefully satisfying following claim for accuracy $ABS(I-\text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL}*ABS(I))$.
H2A3A1	DQAGIE	The routine calculates an approximation result to a given integral $I = \text{Integral of } F \text{ over } (\text{BOUND}, +\text{INFINITY})$ or $I = \text{Integral of } F \text{ over } (-\text{INFINITY}, \text{BOUND})$ or $I = \text{Integral of } F \text{ over } (-\text{INFINITY}, +\text{INFINITY})$, hopefully satisfying following claim for accuracy $ABS(I-\text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL}*ABS(I))$.
H2A3A1	DQAWF	The routine calculates an approximation result to a given Fourier integral $I = \text{Integral of } F(X)*W(X) \text{ over } (A, \text{INFINITY})$ where $W(X) = \cos(\text{OMEGA}*X)$ or $W(X) = \sin(\text{OMEGA}*X)$, Hopefully satisfying following claim for accuracy $ABS(I-\text{RESULT}) \leq \text{EPSABS}$.

H2A3A1	DQAWFE	The routine calculates an approximation result to a given Fourier integral $I = \text{Integral of } F(X)*W(X) \text{ over } (A, \text{INFINITY})$ where $W(X) = \text{COS}(\text{OMEGA}*X)$ or $W(X) = \text{SIN}(\text{OMEGA}*X)$, hopefully satisfying following claim for accuracy $\text{ABS}(I - \text{RESULT}) \leq \text{EPSABS}$.
H2A3A1	QAGI	The routine calculates an approximation result to a given INTEGRAL $I = \text{Integral of } F \text{ over } (\text{BOUND}, +\text{INFINITY})$ OR $I = \text{Integral of } F \text{ over } (-\text{INFINITY}, \text{BOUND})$ OR $I = \text{Integral of } F \text{ over } (-\text{INFINITY}, +\text{INFINITY})$, hopefully satisfying following claim for accuracy $\text{ABS}(I - \text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.
H2A3A1	QAGIE	The routine calculates an approximation result to a given integral $I = \text{Integral of } F \text{ over } (\text{BOUND}, +\text{INFINITY})$ or $I = \text{Integral of } F \text{ over } (-\text{INFINITY}, \text{BOUND})$ or $I = \text{Integral of } F \text{ over } (-\text{INFIN.}, +\text{INFIN.})$, hopefully satisfying following claim for accuracy $\text{ABS}(I - \text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.
H2A3A1	QAWF	The routine calculates an approximation result to a given Fourier integral $I = \text{Integral of } F(X)*W(X) \text{ over } (A, \text{INFINITY})$ where $W(X) = \text{COS}(\text{OMEGA}*X)$ or $W(X) = \text{SIN}(\text{OMEGA}*X)$, Hopefully satisfying following claim for accuracy $\text{ABS}(I - \text{RESULT}) \leq \text{EPSABS}$.
H2A3A1	QAWFE	The routine calculates an approximation result to a given Fourier integral $I = \text{Integral of } F(X)*W(X) \text{ over } (A, \text{INFINITY})$ where $W(X) = \text{COS}(\text{OMEGA}*X)$ or $W(X) = \text{SIN}(\text{OMEGA}*X)$, hopefully satisfying following claim for accuracy $\text{ABS}(I - \text{RESULT}) \leq \text{EPSABS}$.
H2A3A2	DQK15I	The original (infinite integration range is mapped onto the interval (0,1) and (A,B) is a part of (0,1). It is the purpose to compute $I = \text{Integral of transformed integrand over } (A,B)$, $J = \text{Integral of ABS (Transformed Integrand) over } (A,B)$.
H2A3A2	QK15I	The original (infinite integration range is mapped onto the interval (0,1) and (A,B) is a part of (0,1). It is the purpose to compute $I = \text{Integral of transformed integrand over } (A,B)$, $J = \text{Integral of ABS (Transformed Integrand) over } (A,B)$.
I1A1A	DDERKF	Solves initial value problems in ordinary differential equations using a Runge-Kutta-Fehlberg scheme.
I1A1A	DERKF	DERKF solves initial value problems in ordinary differential equations.

I1A1B	DDEABM	Solves initial value problems in ordinary differential equations using an Adams-Bashforth method.
I1A1B	DEABM	Solves initial value problems in ordinary differential equations using an Adams-Bashforth method.
I1A1B	DSTEP2	Integrates a system of first order ODES one step.
I1A1B	STEP2	Integrates a system of first order odes one step.
I1A2	DDEBDF	Solves initial value problems in ordinary differential equations using backward differentiation formulae. It is intended primarily for STIFF problems.
I1A2	DEBDF	Solves initial value problems in ordinary differential equations using backward differentiation formulae. It is intended primarily for STIFF problems.
I1B1	BVSUP	Subroutine BVSUP solves a LINEAR two-point boundary value problem using superposition coupled with an orthonormalization procedure and a variable-step integration scheme.
I1B1	DBVSUP	Subroutine DBVSUP solves a linear two-point boundary value problem using superposition coupled with an orthonormalization procedure and a variable-step integration scheme.
I2B1A1A	HSTCRT	HSTCRT solves the standard five-point finite difference approximation on a staggered grid to the Helmholtz equation in Cartesian coordinates.
I2B1A1A	HSTCSP	HSTCSP solves the standard five-point finite difference approximation on a staggered grid to the modified Helmholtz equation in spherical coordinates assuming axisymmetry (no dependence on longitude).
I2B1A1A	HSTCYL	HSTCYL solves the standard five-point finite difference approximation on a staggered grid to the modified Helmholtz equation in cylindrical coordinates.
I2B1A1A	HSTPLR	HSTPLR solves the standard five-point finite difference approximation on a staggered grid to the Helmholtz equation in polar coordinates.
I2B1A1A	HSTSSP	HSTSSP solves the standard five-point finite difference approximation on a staggered grid to the Helmholtz equation in spherical coordinates and on the surface of the unit sphere (radius of 1).

I2B1A1A	HW3CRT	Subroutine HW3CRT solves the standard seven-point finite difference approximation to the Helmholtz equation in Cartesian coordinates.
I2B1A1A	HWSCRT	Subroutine HWSCRT solves the standard five-point finite difference approximation to the Helmholtz equation in Cartesian coordinates.
I2B1A1A	HWSCSP	Subroutine HWSCSP solves a finite difference approximation to the modified Helmholtz equation in spherical coordinates assuming axisymmetry (no dependence on longitude).
I2B1A1A	HWSCYL	Subroutine HWSCYL solves a standard finite difference approximation to the Helmholtz equation in cylindrical coordinates.
I2B1A1A	HSWPLR	Subroutine HSWPLR solves a finite difference approximation to the Helmholtz equation in polar coordinates.
I2B1A1A	HWSSSP	Subroutine HWSSSP solves a finite difference approximation to the Helmholtz equation in spherical coordinates and on the surface of the unit sphere (radius of 1).
I2B1A2	SEPELI	Automatically discretizes and solves second and (optionally) fourth order finite difference approximations on a uniform grid to the general separable elliptic PDE on a rectangle with any combination of periodic or mixed boundary conditions.
I2B1A2	SEPX4	SEPX4 solves for either the second or fourth order finite difference approximation to the solution of a separable elliptic equation on a rectangle. Any combination of periodic or mixed boundary conditions is allowed. ***SEPX4 is 3 times faster than SEPELI***
I2B4B	BLKTRI	Solves a block tridiagonal system of linear equations (usually resulting from the discretization of separable two-dimensional elliptic equations).
I2B4B	CBLKTR	Subroutine CBLKTR is a complex version of subroutine BLKTRI. It solves a block tridiagonal system of linear equations (usually resulting from the discretization of separable elliptic equations).
I2B4B	CMGNBN	Solves a complex block tridiagonal linear system of equations by a cyclic reduction algorithm.

I2B4B	GENBUN	Solves by a cyclic reduction algorithm the linear system of equations that results from a finite difference approximation to certain 2-d elliptic PDE's on a centered grid.
I2B4B	POIS3D	This subroutine solves three-dimensional block tridiagonal linear systems arising from finite difference approximations to three-dimensional Poisson equations using the Fourier transform package SCLRFFTPAK written by Paul Swarztrauber.
I2B4B	POISTG	Solves a block tridiagonal system of linear equations that results from a staggered grid finite difference approximation to 2-d elliptic PDE's.
J1	FFTD0C	Documentation for FFT package.
J1A1	EZFFTB	A Simplified real, periodic, backward transform.
J1A1	EZFFTF	A simplified real, periodic, forward transform.
J1A1	EZFFTI	Initialize EZFFTF and EZFFTB.
J1A1	RFFTB	Backward transform of a real coefficient array.
J1A1	RFFTF	Forward transform of a real, periodic sequence.
J1A1	RFFTI	Initialize for RFFTF and RFFTB.
J1A2	CFFTB	Unnormalized inverse of CFFTF.
J1A2	CFFTF	Forward transform of a complex, periodic sequence.
J1A2	CFFTI	Initialize for CFFTF and CFFTB.
J1A3	COSQB	Unnormalized inverse of COSQF.
J1A3	COSQF	Forward cosine transform with odd wave numbers.
J1A3	COSQI	Initialize for COSQF and COSQB.
J1A3	COST	Cosine transform of a real, even sequence.
J1A3	COSTI	Initialize for COST.
J1A3	SINQB	Unnormalized inverse of SINQF.
J1A3	SINQF	Forward sine transform with odd wave numbers.

J1A3	SINQI	Initialize for SINQF and SINQB.
J1A3	SINT	Sine transform of a real, odd sequence.
J1A3	SINTI	Initialize for SINT.
K1A1A1	EFC	Fits a piece-wise polynomial curve to discrete data. The piece-wise polynomials are represented as B-splines. The fitting is done in a weighted least squares sense.
K1A1A1	FC	Fits a piece-wise polynomial curve to discrete data. The piece-wise polynomials are represented as B-splines. The fitting is done in a least squares sense. Equality and inequality constraints can be imposed on the fitted curve.
K1A1A2	POLFIT	To fit data in a least squares sense by polynomials in one variable.
K1A2A	LSEI	Solve a linearly constrained least squares problem with equality and inequality constraints, and optionally compute a covariance matrix.
K1A2A	WNNLS	Solve a linearly constrained least squares problem with equality constraints and nonnegativity constraints on selected variables.
K1B1	SCOV	SCOV calculates the covariance matrix for a nonlinear data fitting problem. It is intended to be used after a successful return from either SNLS1 or SNLS1E.
K1B1A1	SNLS1	SNLS1 minimizes the sum of the squares of M nonlinear functions in N variables by a modification of the Levenberg-Marquardt algorithm. This code is a combination of the MINPACK codes (Argonne) LMDER, LMDIF, and LMSTR.
K1B1A1	SNLS1E	SNLS1E is the easy-to-use version of SNLS1 which minimizes the sum of the squares of M nonlinear functions in N variables by a modification of the Levenberg-Marquardt algorithm. This code is the combination of the MINPACK codes (argonne) LMDER1, LMDIF1, and LMSTR.
K6	PVALUE	To use the coefficients generated by POLFIT to evaluate the polynomial fit of degree L, along with the first NDER of its derivatives, at a specified point.
L6A14	RGAUSS	Generates a normally distributed (Gaussian) random number.

L6A21	RAND	Generates a uniformly distributed random number.
L6A21	RUNIF	A portable random number generator.
N6A2A	ISORT	ISORT sorts integer array X and optionally makes the same interchanges in integer array Y. The array X may be sorted in increasing order or decreasing order. A slightly modified QUICKSORT algorithm is used.
N6A2B1	SSORT	SSORT sorts array X and optionally makes the same interchanges in array Y. The array X may be sorted in increasing order or decreasing order. A slightly modified QUICKSORT algorithm is used.
R1	D1MACH	Returns double precision machine dependent constants.
R1	I1MACH	Returns integer machine dependent constants.
R1	R1MACH	Returns single precision machine dependent constants.
R3A	XSETF	Sets the error control flag.
R3B	XSETUA	Sets up to 5 unit numbers to which messages are to be sent.
R3B	XSETUN	Sets output file to which error messages are to be sent.
R3C	NUMKER	Returns most recent error number.
R3C	XERABT	Aborts program execution and prints error message.
R3C	XERCLR	Resets current error number to zero.
R3C	XERCTL	Allows user control over handling of individual errors.
R3C	XERDMP	Prints the error tables and then clears them.
R3C	XERMAX	Sets maximum number of times any error message is to be printed.
R3C	XERROR	Processes an error (diagnostic) message.
R3C	XERRWV	Processes error message allowing 2 integer and two real values to be included in the message.
R3C	XGETF	Returns current value of error control flag.
R3C	XGETUA	Returns unit number(s) to which error messages are being sent.

R3C	XGETUN	Returns the (first) output file to which messages are being sent.
Z	AAAAAA	SLATEC Disclaimer.
Z	DINTRP	Approximates the solution at XOUT by evaluating the polynomial computed in DSTEP2 at XOUT. Must be used in conjunction with DSTEP2.
Z	FDUMP	Symbolic dump (should be locally written).
Z	INTRP	Approximates the solution at XOUT by evaluating the polynomial computed in STEP2 at XOUT. Must be used in conjunction with STEP2.
Z	J4SAVE	Saves and recalls some global variables needed by library error handling routines.
Z	MENU	Contents of the SLATEC library.
Z	S88FMT	Integer to character conversion.
Z	XERPRT	Prints error messages.
Z	XERSAV	Records that an error occurred.

TPLOT

User's Information

TPLOT

Last program update: 22 January 1991
(Added VT240 terminal type)

TPLOT is a general purpose fortran subroutine for plotting data on REGIS, VT340, VT240 or Tektronix terminals. This routine uses the DISSPLA package of standard plotting routines and provides any combination of log/linear axes, automatic or manual scaling, various line and symbol identifiers, annotation (legends, stories and messages), and an ascii or metafile save option so that plots can be edited with a text editor or submitted to a hardcopy output device such as a pen plotter, LN03 printer, or POSTSCRIPT printer. Plot characteristics can be manipulated interactively or pre-defined and passed to the subroutine through named commons.

The subroutine is invoked by a FORTRAN call:

```
CALL TPLOT (NCURV, X, Y, NP, TERMTYPE)
```

NCURV is the number of curves to be plotted on the same graph

X is a concatenated array of abscissas (i.e. for plotting two separate curves A and B with 10 values in A and 5 values in B, the X array would consist of 15 values such that:

```
X(1)  = Xa(1)
      ...
X(10) = Xa(10)
X(11) = Xb(1)
      ...
X(15) = Xb(5)
```

Y is a concatenated array of the corresponding ordinates

NP is an array specifying the number of points for each curve (i.e. for the above example NP(1)=NP_a=10 and NP(2)=NP_b=5.)

TERMTYPE is a three letter character string terminal type designator; 'TEK' for a Tektronix terminal, 'REG' for a REGIS terminal, '240' for a VT240 terminal, or '340' for a VT340 terminal. If TERMTYPE is anything else, the terminal type will be selected automatically by the computer with a default of Tektronix.

When the plot has finished, the user has several options selected by entering a letter code (after hitting <Enter> to get the prompt 'Plot Code [Redraw]: '):

```
AM  create/modify annotation messages
AS  create/modify an annotation story (group of lines)
E   exit
F   cycle through fonts (SIMPLX TRIPLX SWISSL SERIF)
H   help
LB  modify line bolding
LC  modify line colors
LE  create/modify a legend
LS  modify line symbols
LT  modify line types
LX  toggle linear or logarithmic x axis
LY  toggle linear or logarithmic y axis
PS  modify plot size or position
S   save the plot in an ASCII or metafile
```

SH toggle shading of **SWISSL** and **SERIF** fonts
T change the title (the title is printed above the plot)
XL change the abscissa label
XS change the abscissa min, max, step, multiplier
YL change the ordinate label
YS change the ordinate min, max, step, multiplier

When entering text, a null entry (' ') will erase the entire line of text. Alternate character sets (see the **DISSPLA** manual for examples) may be specified by inserting the appropriate changeover character:

] for 'Roman (Default)'
 [for 'Lower Case Greek'
 ~ for 'Upper Case Greek'
 \ for 'Mathematical'
 { for 'Instruction'
 ! for 'Special'

For example, lower case Greek characters may be entered by inserting a [before the equivalent Roman characters (i.e. [L would insert a lower case lambda), and mathematical symbols would be indicated by a \ (i.e. \O would insert the symbol for infinity). The] character always changes back to the default Roman alphabet. As an example of using the instruction character set, superscripting is initiated by entering the six characters {EH.5} and terminated with the six characters {EXHX}. Subscripting is initiated with {LH.5} and terminated with {LXHX}.

Legends are defined by specifying 1) a legend title, 2) lines of text for curve identifiers, 3) character size in inches, and 4) an x,y position relative to axes lengths (i.e. to place the center of the legend in the middle of the plot, XPOS = 0.5 and YPOS = 0.5). Any null (' ') legend entries are skipped and not displayed.

Plot messages are defined by specifying 1) the total number of messages, 2) a line of text (followed by an x,y position relative to axes lengths and an angle) for each message line, and 3) character size in inches.

Plot annotation stories are defined by specifying 1) the number of lines, 2) lines of text, 3) character size in inches, 4) an x,y position relative to axes lengths. Any null (' ') annotation entries are displayed as blank lines.

If either **XS** or **YS** is entered, the user will be asked to enter the desired axis minimum, maximum, increment, and a data multiplier factor. These manually entered values will be held for all subsequent plots until an auto-scale is triggered by entering equal min and max values. The increment is only used when displaying linear axes. If the data multiplier factor is interactively changed, the X or Y labels are modified by appending a "7" to indicate that the corresponding label may also need to be updated.

Predefining Plot Characteristics

All plot characteristics can be predefined in the calling program and passed through named commons after specifying the appropriate variable types and dimensions:

```
CHARACTER*60 TITLE,XLBL,YLBL,LEGTIT*20,LEGTXT(50),STYTXT(50),MESTXT(50)
```

Only those commons pertaining to the predefined features must be included in the calling program. Any plot feature whether predefined or not may be interactively changed once the plot is displayed.

COMMON/PLOTLINES/LBOLD(50),LCOLOR(50),LTYPE(50),LSYMBOL(50),LSKIP(50),SIZSYM

		<u>REGIS</u>	<u>HP Plotter-Pen</u>	<u>VT340</u>
LBOLD = 0 for normal line	LCOLOR = 0	red	black-1	white
LBOLD = 1 for bold line	LCOLOR = 1	red	red-2	red
	LCOLOR = 2	green	green-3	green
	LCOLOR = 3	blue	blue-4	blue
	LCOLOR = 4	red	magenta-5	magenta
	LCOLOR = 5	red	yellow-6	yellow
	LCOLOR = 6	red	cyan-7	cyan
	LCOLOR = 7	red	white-8	white

(i.e. if the first line is to be bold and blue then LBOLD(1)=1 and LCOLOR(1)=3).

LTYPE is an array specifying the display mode for each curve:

LTYPE = -2 for symbols only
 LTYPE = -1 for both line and symbols
 LTYPE = 0 for solid line
 LTYPE = 1 for dashed line
 LTYPE = 2 for dotted line
 LTYPE = 3 for chain-dashed line
 LTYPE = 4 for chain-dotted line

LSYMBOL is an array specifying the symbol (if used) for each curve. There are a total of 18 symbols which are described in the DISSPLA documentation for calls to MARKER. Some of the symbols are listed below:

LSYMBOL = 0 for 'Square'
 LSYMBOL = 1 for 'Hexagon'
 LSYMBOL = 2 for 'Triangle'
 LSYMBOL = 3 for 'Cross'
 LSYMBOL = 4 for 'X'
 LSYMBOL = 6 for 'Inverted Triangle'
 LSYMBOL = 7 for 'Square' + 'X'
 LSYMBOL = 13 for 'Circle' + 'X'
 LSYMBOL = 15 for 'Filled Circle'
 LSYMBOL = 16 for 'Circle'

LSKIP is the symbol skip factor (i.e. 2 means plot a symbol at every 2nd point).
SIZSYM is the symbol size in inches.

COMMON/PLOTSTORY/NSTYLS, SIZSTY, STYTXT, XSTYF, YSTYF

NSTYLS is the number of annotation story lines
SIZSTY is the annotation (story) character height in inches
STYTXT is the array of text lines
XSTYF is the x position of the story center relative to the X axis
YSTYF is the y position of the story center relative to the Y axis

COMMON/MESSAGE/NMESLNS, SIZMES, MESTXT, XMESF(50), YMESF(50), ANGME(50)

NMESLNS is the number of messages
SIZMES is the annotation line character height in inches
MESTXT is the array of messages
XMESF is the x position of the message center relative to the X axis

YMESF is the y position of the message center relative to the Y axis
ANGMES is the angle of the message

COMMON/PLOTLEGEND/XLEGF, YLEGF, SIZELEG, LEGTIT, LEGTXT

XLEGF is the x position of legend center relative to the X axis
YLEGF is the y position of legend center relative to the Y axis
SIZELEG is the legend character height in inches
LEGTIT is the legend title
LEGTXT is the array of legend lines

COMMON/PLOTLABEL/XTITF, YTITF, SIZTIT, TITLE, SIZLAB, XLBL, YLBL, IFONT, ISHAD

XTITF is the x position of the title center relative to the X axis
YTITF is the y position of the title center relative to the Y axis
SIZTIT is the title character height in inches
TITLE is the title
SIZLAB is the character height for the X and Y axes in inches
XLBL is the X label
YLBL is the Y label
IFONT specifies the font (1=SIMPLX, 2=TRIPLX, 3=SWISSL, 4=SERIF)
ISHAD specifies shading for the SWISSL and SERIF fonts

**COMMON/PLOTAXES/IXS, XMN, XSTEP, XMX, XMULT, LOGX,
IYS, YMN, YSTEP, YMX, YMULT, LOGY, IGRID**

IXS = 1 to specify no autoscaling of X data
XMN is the minimum X axis value
XSTEP is the X axis major tic mark increment
XMX is the maximum X axis value
XMULT = multiply X values by XMULT
LOGX = 1 to specify a logarithmic X axis
IYS = 1 to specify no autoscaling of Y data
YMN is the minimum Y axis value
YSTEP is the Y axis major tic mark increment
YMX is the maximum Y axis value
YMULT = multiply Y values by YMULT
LOGY = 1 to specify a logarithmic Y axis
IGRID = 1 specifies a plot grid

COMMON/PLOTSIZE/XORIGIN, YORIGIN, XLENIN, YLENIN

XORIGIN, YORIGIN specifies the plot origin in inches
XLENIN specifies the x axis length in inches
YLENIN specifies the y axis length in inches

Link Instructions

The following link command is required to use the TPLLOT subroutine:

LINK user_program, 'TPLLOT'

where **user_program** is the user's object code containing the call to TPLLOT and **TPLLOT** is a predefined system symbol pointing to the TPLLOT and DISSPLA routines.

VMS Kermit User's Information

Table of Contents

1. VAX/VMS KERMIT

- 1.1 The VAX/VMS File System**
- 1.2 Program Operation**
- 1.3 Conditioning Your Job for Kermit**
- 1.4 Kermit-32 Commands**
 - 1.4.1 Commands for File Transfer**
 - 1.4.2 Server Operation**
 - 1.4.3 Commands for Local File Management**
 - 1.4.4 The CONNECT Command**
 - 1.4.5 The SET and SHOW Commands**
 - 1.4.6 Program Management Commands**
- 1.5 Raw Upload and Download**
- 1.6 Installation of Kermit-32**

1. VAX/VMS KERMIT

Authors: Robert C. McQueen, Nick Bush, Stevens Institute of Technology;
Jonathan Welch, University of Massachusetts;
Burt Johnson, Diversified Computer Systems, Inc.

Documentation: C. Gianone, F. da Cruz, Columbia University
with thanks to the program's authors
This documentation was reformatted by S. L. Steely

Language: Bliss-32
Version: 3.3.126
Date: July 1990

VAX/VMS Kermit-32 Capabilities At a Glance:

Local operation:	Yes
Remote operation:	Yes
Transfers text files:	Yes
Transfers binary files:	Yes
Wildcard send:	Yes
Long packets:	Yes
Sliding windows:	No
Attribute packets:	No
File transfer interruption:	Yes
Filename collision avoidance:	Yes
Timeouts:	Yes
8th-bit prefixing:	Yes
Repeat character compression:	Yes
Alternate block check types:	Yes
Communication settings:	Yes
Transmit BREAK:	Yes
IBM mainframe communication:	Yes
Transaction logging:	Yes
Session logging (raw capture):	Yes
Debug logging:	Yes
Raw transmit:	Yes
Act as server:	Yes
Talk to server:	Yes
Advanced commands for servers:	Yes
Local file management:	Yes
Initialization file:	Yes (VMSKERMIT.INI)
Command Macros:	No
Script programming language:	No
International Character Sets:	No

Kermit-32 is a program that implements the Kermit file transfer protocol for the Digital Equipment Corporation VAX series computers under the VAX/VMS operating system. It is written in BLISS-32 and MACRO-32, with sources for all BLISS modules also available as MACRO-32 sources. Kermit-32 should run on any VAX/VMS system from version 4.0 on (Version 3.1 of Kermit-32 is the last version that runs under pre-4.0 releases of VMS).

The first section of this chapter will describe the things you need to know about the VAX/VMS file system and how Kermit-32 uses it. The second section describes the special features of Kermit-32. The final section contains information of interest to those who need to install Kermit-32 on a system.

1.1 The VAX/VMS File System

The two main items of interest of the VAX/VMS file system (for the Kermit user) are the format of file specifications and the types of files and file data.

VAX/VMS File Specifications

VAX/VMS file specifications are of the form

NODE::DEVICE:[DIRECTORY]NAME.TYPE;VERSION

Under version 4.0 and later of VMS, NAME, TYPE and each item in DIRECTORY may be up to 39 characters long, and may contain alphanumeric characters plus underscore. Under earlier versions, NAME and DIRECTORY could be at most 9 characters each and TYPE could be at most 3.

VERSION is a decimal number indicating the version of the file (generation).

DEVICE may be either a physical or logical device name. If it is a logical name, it may be up to 63 characters long and may contain alphanumeric characters plus dollar signs and underscores. NODE may be either a logical name which translates to a DECnet node name or a physical DECnet node name. In either case, access information can be included (see the DECnet-VMS User's guide for more information). The node name is not normally present, since most files are accessed on the same node where the user's job is running. The version number is not normally given (in fact, should not normally be given). When device and/or directory are not supplied, they default to the user's current default device and directory. Therefore, NAME.TYPE is normally all that is needed to specify a file on the user's

default device and directory. This is also all that Kermit-32 will normally send as the name of a file being transferred.

The node field specifies the name (and access information) for the DECnet node where the file is located. Kermit-32 does not transmit the node field to the target system, but will attempt to honor a node field in an incoming file name.

The device field specifies a physical or "logical" device upon which the file is resident. The directory field indicates the area on the device, for instance the area belonging to the owner of the file. Kermit-32 does not normally transmit the device or directory fields to the target system, but will attempt to honor device or directory fields that may appear in incoming file names. It will not create new directories, however, so any directory specified in an incoming filename must already exist.

The name is the primary identifier for the file. The type, also called the "extension", is an indicator which, by convention, tells what kind of file we have. For instance FOO.FOR is the source of a Fortran program named FOO; FOO.OBJ might be the relocatable object module produced by compiling FOO.FOR; FOO.EXE would be an executable program produced by LINKing FOO.OBJ, and so forth.

VAX/VMS allows a group of files to be specified in a single file specification by including the special "wildcard" characters, "*" and "%". A "*" matches any string of characters, including no characters at all; a "%" matches any single character. Here are some examples:

***.FOR** = all files of type FOR (all Fortran source files) in the default directory.

FOO.* = files of all types with name FOO.

F*.* = All files whose names start with F.

F*X*.* = All files whose names start with F and contain at least one X.

%.* = All files whose names are exactly one character long.

.%% = All files whose types are at least two characters long.

Wildcard notation is used on many computer systems in similar ways, and it is the mechanism most commonly used to instruct Kermit to send a group of files.

TEXT FILES AND BINARY FILES

The file system used by VAX/VMS provides for a large number of attributes to be associated with each file. These attributes provide some indication of whether the file is a text file, or is some other type of non-text data. The two major attributes that affect VMS Kermit are the record type and record attribute. The record type describes how logical records are stored in the file. Records may be of some fixed length (specified by another attribute), or variable length (specified within each record), or stream (implying that records -- if there are any -- are separated by control characters within the data). The record attributes describe how the breaks between records are to be treated. For example, a record attribute of implied carriage return means that any program reading the file with intentions of printing it should add a carriage return / linefeed sequence between each record. Other record attributes include FORTRAN carriage control and print file format.

The most common method of storing text in a file under VAX/VMS is to store one line of text per record (variable length records), with a carriage return / linefeed sequence implied by the end of the record (implied carriage return). This is the method Kermit-32 uses to store files it receives when using FILE TYPE ASCII (text). Other formats are also used to store text under VAX/VMS, but the one used by Kermit-32 is the only one that is handled correctly by all known utility programs under VAX/VMS. Also, most programs which work with text files (the editor EDT, for example) place some limit on the length of the lines which can be handled. Typically this is 255. Kermit-32 can write text files with up to 4095 characters on a line, which means a text file from another system may be transferred and stored correctly by Kermit-32, but may still be unusable by certain VAX/VMS programs.

Certain PC applications may create text files with lines even longer than Kermit-32's maximum. Typical examples are the ASCII export procedures of database, spreadsheet, and CAD packages. If you try to send such a file to Kermit-32, the transfer will fail with a message:

%KERMIT32-E-REC_TOO_BIG, Record too big for KERMIT's internal buffer

If this happens, you can SET FILE TYPE BINARY on the VAX before transferring the file to it. You should still be able to use the file as a text file, with the above proviso about record length.

There is no standard format for storing binary files. In general, any record format with no record attributes can be used for binary files. Since programs which work with binary files under VAX/VMS expect to see some particular format, more information is needed for transfer of binary files than for text

files. The current version of Kermit-32 is not capable of transferring all types of binary files which were created on a VAX/VMS system to another system and retrieving them intact, nor is it capable of transferring all of types binary files created on a VAX/VMS system to another VAX/VMS, P/OS, or RSX-11M/M+ system intact. However, certain formats of binary files can be transferred, and binary files from some other systems may be transferred to a VAX and recovered intact. See the section on the SET FILE command for details.

Using two programs supplied with Kermit-32, it is possible to transfer almost any type of file between VAXes, or between a VAX and a P/OS or RSX-11M/M+ system. The VMSHEX program converts a binary file to text (using a variation on Intel hex format). The resulting file can be transferred as an ordinary text file, and finally "dehexified" on another VMS system using the VMSDEH program to reproduce the original file with all the attributes intact. Since these text files are about twice the size of the original binary files, the transfers take longer. On the plus side, the text versions of the files can be transferred to any system with a Kermit and still retrieved intact. They can also be transferred over 7-bit data paths without any problems. The Kermit-32 installation procedure (described later) makes use of hexified versions of the Kermit-32 binary executable file and VMSDEH to restore it as a binary .EXE file, or task image.

USING THE VAX TO ARCHIVE MICROCOMPUTER FILES

You can use Kermit to send textual files from a microcomputer or any 8-bit computer system to VAX/VMS with no special provisions, since Kermit-32 stores incoming files as text files (variable length records with implied carriage returns) unless it is explicitly told otherwise. But Kermit-32 has no automatic way of distinguishing an incoming binary file from an incoming text file. You must inform Kermit-32 that a file it is about to receive is to be stored as a binary file. This is done using the SET FILE TYPE BINARY command. This instructs Kermit-32 to store the data it receives in the file without checking for carriage return, line feed sequences. The file it creates will be variable record length, with no record attributes. Each record will contain 510 bytes of data, except the last, -which will contain whatever amount is left before the end of the file. This allows Kermit-32 to correctly return exactly the data it was sent when the file is returned to the original system.

Note that because of the need to use a different file type for binary files, it is not possible to use a "wildcard send" command to send a mixture of text and binary files to a VAX system unless the text files are not intended for use on the VAX; rather, you must send all text files with Kermit-32's file type set to ASCII, and all binary files with the file type set to binary.

Once you get the foreign file into the VAX system, stored with the correct file type, you need take no special measures to send it back to its system of origin. This is because Kermit-32 honors the record type and attributes of the file as it is stored on the VAX. In fact, SET FILE TYPE BINARY, ASCII, or FIXED only affects how Kermit-32 receives files -- it does not affect how Kermit-32 transmits files.

FILES KERMIT-32 CANNOT HANDLE

The Kermit protocol can only accommodate transfer of sequential files, files which are a linear sequence of bytes (or words).

Some files on a VAX/VMS system are not sequential, and cannot be successfully sent or received by Kermit-32. These are mainly indexed data files, but can also include other files which require more than just the data in the file to be completely reconstructed. External control information and file attributes are not transmitted. However, any VMS file can be transferred with Kermit if it has been "hexified" with VMSHEX.

1.2 Program Operation

If a system-wide foreign command "kermit" is defined for Kermit-32 (see section on installation), then you can run the program by just typing its name:

```
$ kermit
```

If you get a message like:

```
%DCL-W-IVVERB, unrecognized command verb - check validity and spelling
```

then Kermit has not been installed properly on your VMS system. If you know where the KERMIT.EXE file is stored (for example, in the SYS\$SYSTEM: area) then you can define a "kermit" command for yourself by including a line like this in your LOGIN.COM file:

```
kermit := $sys$system:kermit.exe
```

When you invoke Kermit by name only, it will enter interactive prompting mode, and allow you to type repeated commands until you exit with the EXIT command. Kermit-32's normal prompt is "Kermit-32>".

Kermit-32 will also accept a single command on the command line, like this:

\$ Kermit send foo.bar

In this case, the program will exit immediately after executing the single command.

In either case, Kermit reads and executes commands from its "initialization file", VMSKERMIT.INI, if any, in your login directory before it executes any other commands.

Kermit-32 will try to open the file VMSKERMIT with a default filetype of .INI. If the logical name VMSKERMIT exists then an attempt to open the file pointed to by the value of that logical name will be made instead. (For example, some sites might wish to set site-wide default parameters in a system-wide Kermit-32 initialization file, and so they might define the system-wide logical name VMSKERMIT to point at such a file. The last command in this file could be TAKE SYS\$LOGIN:VMSKERMIT.INI to "chain" to the user's initialization file.

Command keywords may be abbreviated to their shortest prefix that sets them apart from any other keyword valid in that field.

1.3 Conditioning Your Job for Kermit

Kermit-32 does as much as it can to condition your line for file transfer. It saves all your terminal settings, and restores them after use. However, there are some sources of interference over which Kermit-32 has no control. In particular, messages issued by other processes in your job could become mingled with Kermit packets and slow things down or stop them entirely. This is a fairly rare occurrence and can be easily avoided by not running any other process which wishes to perform I/O to your terminal while you are running Kermit-32.

Normally, when Kermit-32 is run, it assumes you wish to use it in remote mode and perform file transfers over the terminal line which controls your job. This can be overridden, however, by defining a logical name which equates to some other terminal line in the system. The default terminal line to be used for file transfers is determined by the first of the following logical names which translates to a terminal line which is available for use by your process: KER\$COMM, SYS\$INPUT, SYS\$OUTPUT, and SYS\$COMMAND. If none of these logical names translate to an available terminal line, there is no default terminal line and a SET LINE command must be used before any transfer command is performed. Note that this is the typical case in a batch job.

Kermit-32 will also default the type of parity to be used on the communication line to that which is set on its default terminal line when it is started. This means that if all communication at a site is normally done using even parity (for example), Kermit-32 will also use even parity. If you need to use another kind of parity, use the SET PARITY command to change it.

There are two things to keep in mind when using Kermit-32 in local mode (where the file transfers are done over a different terminal line from where commands are typed):

- Under VAX/VMS, every terminal line has an owner UIC and protection code associated with it. This UIC and protection is used to determine who can allocate (and therefore use) the terminal line when they are not logged in on that line. Therefore, in order for Kermit-32 to be able to perform file transfers over a terminal line other than the one on which you are logged in, the field of the protection code for the terminal which applies to your job (based on your UIC and the owner UIC of the terminal) must allow your job access to the terminal. You may need to request your system manager to change the protection for a terminal line to allow you to use it with Kermit-32 in local mode. See the section on Installation for details.
- Terminal lines which have been declared as modem control lines will have the phone "hung up" whenever the terminal line becomes free (deallocated). This means that if you do not use the DCL ALLOCATE command to allocate the terminal line to your job before entering Kermit-32, exiting Kermit-32 will cause the terminal line to "hang up" the modem. If you do wish to get to DCL after having used Kermit-32 to connect a modem control line which you do not have allocated, you can use the PUSH command to spawn a subprocess running DCL, so that Kermit will keep the connection active.

1.4 Kermit-32 Commands

Kermit-32 has the following commands:

@ synonym for "take".
BYE to remote server.
CONNECT as terminal to remote system.
EXIT from Kermit-32.
FINISH Shut down remote server.

GET remote files from server.
HELP with Kermit-32.
LOCAL prefix for local file management commands.
LOG remote terminal session.
LOGOUT remote server.
PUSH to DCL command level.
QUIT from Kermit-32.
RECEIVE files from remote Kermit.
REMOTE prefix for remote file management commands.
SEND files to remote Kermit.
SERVER mode of remote operation.
SET various parameters.
SHOW various parameters.
STATUS about most recent file transfer.
TRANSMIT Transmit (upload) a file with no error checking.
TAKE Kermit-32 commands from a file.

1.4.1 Commands for File Transfer

Kermit-32 provides the standard SEND, RECEIVE, and GET commands for transferring files using the Kermit protocol.

THE SEND COMMAND

Syntax: SEND filespec

The SEND command causes a file or file group to be sent from the VAX to the other system. If filespec contains wildcard characters then all matching files will be sent, in alphabetical order (according to the ASCII collating sequence) by name. If filespec does not contain any wildcard characters, then the single file specified by filespec will be sent.

Only the most recent generation of a file is sent unless the file specification includes specific or wild generation numbers.

Files will be sent with at least their VAX/VMS file name and type (for instance FOO.BAR). If a SET FILE NAMING FULL command has been given, Kermit-32 will also send the device name, directory name and version number (for instance USER\$DISK:[JOE]FOO.BAR;25). If a SET FILE NAMING UNTRANSLATED command has been given, Kermit-32 will send the file name, type and version number (for instance FOO.BAR;25). If a SET

FILE NAMING NORMAL_FORM command has been given (this is the initial default), Kermit-32 will only send the file name and type.

Each file will be sent according to the record type and attributes recorded in its file descriptor. Kermit-32 attempts to translate all text file record formats (including those with FORTRAN or print carriage control) to a format usable on any system. Note that there is no need to set the **FILE TYPE** parameter for sending files, since Kermit-32 always uses the information from the file descriptor to determine how to send the file.

If communication line parity is being used (see **SET PARITY**), Kermit-32 will request that the other Kermit accept a special kind of prefix notation for files that contain 8-bit data. This is an optional Kermit protocol feature, supported by most modern Kermit programs. If the other Kermit does not agree to use this feature, binary files cannot be sent correctly. This includes executable programs (like .EXE files, CP/M .COM files), relocatable object modules (.OBJ files), as well as any text file containing characters with the eighth bit on.

Kermit-32 will also ask the other Kermit whether it can handle a special prefix encoding for repeated characters. If it can, then files with long strings of repeated characters will be transmitted very efficiently. Columnar data, highly indented text, and binary files are the major beneficiaries of this technique.

If you're running Kermit-32 in local mode, for instance dialing out from a VAX to another system using an autodialer, you should have already run Kermit on the remote system and issued either a **RECEIVE** or a **SERVER** command. Once you give Kermit-32 the **SEND** command, the name of each file will be displayed on your screen as the transfer begins. If the file is successfully transferred, you will see "[OK]", otherwise there will be an error message.

During local operation, you can type Control-A at any point during the transfer to get a brief status report. You may also type Control-X or Control-Z to interrupt the current file or file group.

THE RECEIVE COMMAND

Syntax: **RECEIVE** [filespec]

The **RECEIVE** command tells Kermit-32 to receive a file or file group from the other system. If only one file is being received, you may include the optional filespec as the name to store the incoming file under; otherwise, the

name is taken from the incoming file header. If the name in the header is not a legal VAX/VMS file name, Kermit-32 will normally replace the illegal characters with "X" (see SET FILE NAMING NORMAL_FORM).

If an incoming file has the same name as an existing file, Kermit-32 just creates a new version of the same name and type, for instance FOO.BAR;3, FOO.BAR;4.

Incoming files are stored with the prevailing file type, ASCII by default, which is appropriate for text files. If you are asking Kermit-32 to receive binary files from a microcomputer or other 8-bit system, you must first type SET FILE TYPE BINARY. Otherwise, an error may occur when receiving the file, or carriage return / linefeed sequences will be added to the file, making it useless when sent back to the system of origin.

If parity is being used on the communications line, then 8th-bit prefixing will be requested. If the other side cannot do this, binary files cannot be transferred correctly. If parity is being added externally to Kermit and VMS (for example by some kind of communication device, or a public data network like Telenet) then you must inform Kermit-32 about it using the SET PARITY command, or else Kermit-32 will not know that it has to do 8th-bit prefixing, and the file transfer will fail.

If an incoming file does not arrive in its entirety, Kermit-32 will normally discard it; it will not appear in your directory. You may change this behavior by using the command SET INCOMPLETE KEEP, which will cause as much of the file as arrived to be saved in your directory.

If you are running Kermit-32 in local mode, you should already have issued a SEND command to the remote Kermit, and then escaped back to Kermit-32. As files arrive, their names will be displayed on your screen. You can type Control-A during the transfer for a brief status report.

If a file arrives that you don't really want, you can attempt to cancel it by typing Control-X; this sends a cancellation request to the remote Kermit. If the remote Kermit understands this request (not all implementations of Kermit support this feature), it will comply; otherwise it will continue to send. If a file group is being sent, you can request the entire group be cancelled by typing Control-Z.

THE GET COMMAND

Syntax: GET [remote-filespec]

The GET command requests a remote Kermit server to send the file or file group specified by remote-filespec. This command can be used only when Kermit-32 is in local mode, with a Kermit server on the other end of the line specified by SET LINE. This means that you must have CONNECTed to the other system, logged in, run Kermit there, issued the SERVER command, and escaped back to the VAX.

The remote filespec is any string that can be a legal file specification for the remote system; it is not parsed or validated locally. Any leading spaces before the remote filespec are stripped, and lower case characters are raised to upper case.

As files arrive, their names will be displayed on your screen. As in the RECEIVE command, you may type Control-A to get a brief status report, Ctrl-X to request that the current incoming file be cancelled, Ctrl-Z to request that the entire incoming batch be cancelled.

If the remote Kermit is not capable of server functions, then you will probably get an error message back from it like "Illegal packet type". In this case, you must connect to the other Kermit, give a SEND command, escape back, and give a RECEIVE command.

THE STATUS COMMAND

Give statistics about the most recent file transfer.

THE PUSH COMMAND

Syntax: PUSH

Spawn a DCL subprocess, to which you may issue any DCL commands. Type LOGOUT to return to Kermit-32.

THE TAKE COMMAND

Syntax: TAKE file-spec [/DISPLAY]

Where 'file-spec' is any normal VAX/VMS file specification. If file-spec does not specify a file-type Kermit-32 will supply a default of .COM. The /DISPLAY option causes the commands read from the file to be displayed on the user's terminal.

The **TAKE** command tells Kermit-32 to execute commands from the specified file. You may also use the VMS notation "@" instead of **TAKE** to specify a command file.

If it exists, the file **VMSKERMIT.INI** (or, if the logical name **VMSKERMIT** is defined, whatever file it points to) is automatically taken upon program startup.

1.4.2 Server Operation

THE SERVER COMMAND

The **SERVER** command puts a remote Kermit-32 in "server mode", so that it receives all further commands in packets from the local Kermit. The Kermit-32 server is capable (as of this writing) of executing the following remote server commands: **SEND**, **GET**, **FINISH**, **BYE**, **REMOTE DIRECTORY**, **REMOTE CWD**, **REMOTE SPACE**, **REMOTE DELETE**, **REMOTE TYPE**, **REMOTE HELP**, **REMOTE COPY**, **REMOTE RENAME**, **REMOTE SEND_MESSAGE**, **REMOTE WHO**, and **REMOTE HOST**.

Any nonstandard parameters should be selected with **SET** commands before putting Kermit-32 into server mode, in particular the file type. The Kermit-32 server can send all files in the correct manner automatically. However, if you need to ask Kermit-32 to receive binary files you must issue the **SET FILE TYPE BINARY** command before putting it into server mode, and then you must only send binary files. You cannot send a mixture of text files and 8-bit binary files to a Kermit-32 server unless the files are not for use on the VAX.

COMMANDS FOR SERVERS

When running in local mode, Kermit-32 allows you to give a wide range of commands to a remote Kermit server, with no guarantee that the remote server can process them, since they are all optional features of the protocol. Commands for servers include the standard **SEND**, **GET**, **BYE**, **LOGOUT** and **FINISH** commands, as well as the **REMOTE** command.

Syntax: REMOTE command

Send the specified command to the remote server. If the server does not understand the command (all of these commands are optional features of the

Kermit protocol), it will reply with a message like "Unknown Kermit server command". If does understand, it will send the results back, and they will be displayed on the screen. The REMOTE commands are:

COPY filespec Copy file. The server is asked to make a copy of the specified file. Kermit-32 will prompt for the new file name on a separate line. Both filespecs must be in the correct format for the remote system. Kermit-32 does not parse or validate the file specifications. Any leading spaces will be stripped and lower case characters converted to upper case. Note that this command simply provides for copying a file within the server's system - it does not cause a file to be transferred.

CWD [directory] Change Working Directory. If no directory name is provided, the server will change to the default or home directory. Otherwise, you will be prompted for a password, and the server will attempt to change to the specified directory. The password is entered on a separate line, and does not echo as you type it. If access is not granted, the server will provide a message to that effect. Note that while not all server Kermits require (or accept) a password to change the working directory, Kermit-32 will always ask for one when a directory name is provided.

DELETE filespec Delete the specified file or files. The names of the files that are deleted will appear on your screen.

DIRECTORY [filespec]

The names of the files that match the given file specification will be displayed on your screen, perhaps along with size and date information for each file. If no file specification is given, all files from the current directory will be listed.

DISK_USAGE [directory]

Display information about disk usage in the given directory (or by the given user). If no directory is provided, disk usage information is provided for the current working directory (or user). This is the same as the REMOTE SPACE command.

EXIT Requests the server to leave Kermit, allowing the terminal to be used for normal commands.

FINISH Requests the server to return to the Kermit prompt, allowing statistics to be obtained about the transfers.

HELP [topic] Provide information about the given topic. If no topic is given, provide a list of the functions that are available from the server. Some servers may ignore the topic and always display the same information.

HOST [command] Pass the given command to the server's host command processor, and display the resulting output on your screen.

LOGIN user-id Supply information to the server Kermit to indicate what user-id, account and password are to be used. The server Kermit may use this to validate the user's access to the system as well as for billing purposes. It may also use this information to provide the user with access to files on its system.

LOGOUT Request the server to exit Kermit and logout its job (or process). This command is identical to the LOGOUT command.

RENAME filespec Change the name on the specified file (or files). Kermit-32 will prompt for the new file specification on the next line. Both file specifications must be valid for the server's system.

SEND_MESSAGE destination-address
Request the server to send a single line message to the specified destination address (which might be a user-id, terminal designator, or some other item, depending upon the server Kermit). Kermit-32 will prompt for the single line message on the next line.

SPACE [directory]
Display information about disk usage in the given directory (or by the given user). If no directory is provided, disk usage information is provided for the current working directory (or user). This is the same as the REMOTE DISK_USAGE command.

STATUS Display information about the status of the server.

TYPE filespec Display the contents of the specified file on your screen.

WHO [user-id] Display information about the given user. If no user-id is given, display information about the currently active users. Kermit-32 will prompt for options for selecting what information to display and/or formatting parameters. The format of both the user-id and the options are dependent upon the server Kermit.

1.4.3 Commands for Local File Management

Syntax: LOCAL [command]

Execute the specified command on the local system -- on the VAX/VMS system where Kermit-32 is running. These commands provide some local file management capability without having to leave the Kermit-32 program. These commands are very similar to the REMOTE commands in function and syntax. They are all executed locally, and are available when Kermit-32 is either local or remote. The arguments to these commands are the same as the arguments expected from the user Kermit when Kermit-32 is processing a command in server mode.

COPY filespec Make a copy of the given file (or files). Kermit-32 will prompt for the new file specification. The command is actually performed by using the DCL COPY command (COPY/LOG old-file new-file), and any options which are valid on the DCL COPY command may be included.

CWD [directory] Change working directory, or, in VAX/VMS terminology, change the default device/directory. This command takes the same arguments as the DCL SET DEFAULT command (i.e., a device and directory, only a directory, or only a device). If no argument is given, the default device and directory are reset to that in effect when Kermit-32 was run. The new default device and directory will be typed out.

DELETE filespec Delete the specified file or files. This command is performed by using the DCL DELETE command (DELETE/LOG filespec). Therefore, any options which are valid on the DCL DELETE command may be included.

DIRECTORY [filespec]

Provide a directory listing of the specified files. This command is performed by using the DCL DIRECTORY command (DIRECTORY filespec), so any options valid for the DCL DIRECTORY command may be included.

DISK_USAGE [uic]

Display disk usage information for the given UIC. If no UIC is given, display disk usage information for the process UIC. This command is performed by using the DCL SHOW QUOTA command (SHOW QUOTA or SHOW QUOTA/USER=uic).

HELP Display the help message describing the server commands which are available.

HOST DCL-command

Perform the given DCL command. The command should not perform any action which will require more input. Any output resulting from the command will be typed on the terminal.

RENAME filespec Change the name of the specified file. Kermit-32 will prompt for the new name on the next line. This command is performed by using the DCL RENAME command (RENAME/LOG old-file new-file), so any options which are valid on the DCL RENAME command may be included.

SEND_MESSAGE terminal-name

Send a single line message to the given terminal. Kermit-32 will prompt for the message on the next line. Since this command is performed using the DCL REPLY command

REPLY/TERMINAL=terminal-name "message"

OPER privileges are needed to perform it.

TYPE filespec Display the contents of the specified file or files at your terminal. Each file will be preceded by its name in angle brackets.

1.4.4 The CONNECT Command

Syntax: CONNECT [terminal-name]

Establish a terminal connection to the system connected to the terminal line specified here or in the most recent SET LINE command, using the currently set communication parameters (local-echo, parity, etc). Get back to Kermit-32 by typing the escape character followed by the letter C. The escape character is Control-Rightbracket (^]) by default. When you type the escape character, several single-character commands are possible:

- B** Send a BREAK signal.
- C** Close (but do not hang up) the connection and return to Kermit-32.
- Q** If a session log is active, temporarily Quit logging.
- R** Resume logging to the session log.
- S** Show status of the connection.
- 0** Send a null character.

? List all the possible single-character arguments.
^] (or whatever you have set the escape character to be):
Typing the escape character twice sends one copy of it to the connected host.

You can use the SET ESCAPE command to define a different escape character, and SET PARITY, and SET LOCAL_ECHO to change those communication-line-oriented parameters. Type the SHOW LINE command for information about your current communication settings.

Kermit-32 does not have any special autodialer interface. It assumes that the connection has already been made and the line assigned. If the line has an autodialer attached to it, then you can type commands to the autodialer after you CONNECT.

1.4.5 The SET and SHOW Commands

THE SET COMMAND

Syntax: SET parameter [option [value]]

Establish or modify various parameters for file transfer or terminal connection. You can examine their values with the SHOW command. The following parameters may be SET:

BLOCK_CHECK Packet transmission error detection method
DEBUGGING Record or display state transitions or packets
DELAY How long to wait before starting to send
ESCAPE Character for terminal connection
FILE For setting file parameters like file type
HANDSHAKE For establishing half duplex line turnaround handshake
IBM_MODE For communicating with an IBM mainframe
INCOMPLETE_FILE What to do with an incomplete file
LINE Terminal line to use for file transfer or CONNECT
LOCAL_ECHO For terminal connection, ON or OFF
MESSAGE The type of timeout to be done during transfers
PARITY Character parity to use
PROMPT Change the program's command prompt
RECEIVE Various parameters for receiving files
REPEAT_QUOTE Character to use for repeat compression
RETRY How many times to retry a packet before giving up

SEND Various parameters for sending files
TRANSMIT Control **TRANSMIT** command echo and delay
SET DEBUGGING

Syntax: **SET DEBUGGING** options

Record the packet traffic, either on your terminal or in a file. Some reasons for doing this would be to debug a version of Kermit that you are working on, to record a transaction in which an error occurred for evidence when reporting bugs, or simply to vary the display you get when running Kermit-32 in local mode. Options are:

- ON** Display each incoming and outgoing packet (lengthy).
- OFF** Don't display or record debugging information (this is the normal mode). If debugging was in effect, turn it off.

The debugging information is recorded in the file specified by the most recent **LOG DEBUGGING** command.

SET ESCAPE

SET ESCAPE octal-number

Specify the control character you want to use to "escape" from remote connections back to Kermit-32. The default is 35 (Control-J). The number is the octal value, 1 to 37 (or 177), of the ASCII control character you want to use, for instance 2 is Control-B.

SET FILE

Syntax: **SET FILE** parameter keyword

Establish file-related parameters:

BLOCKSIZE number

Specify the record size for incoming files when the file type is set to **BINARY**, **FIXED**, or **BLOCK**. Note that "blocksize" is a misnomer, but one which is commonly used in VMS. All VMS disk files have a true blocksize of 512 bytes. The Kermit "blocksize" (as well as the blocksize referred to in many VMS commands, like **BACKUP**, and help files) is really the record size.

TYPE keyword

How Kermit-32 should treat and store the file that is being sent to it, i.e. that Kermit-32 is receiving, and (in the case of FILE TYPE BLOCK only) how it is to read a file it is sending from disk. The choices are ASCII, BINARY, BLOCK, and FIXED. The BINARY, BLOCK, and FIXED types use a default record size (described below) which may be overridden, for received files only, with the SET FILE BLOCKSIZE command. Because the VMS file system is so complex, and because files created by different applications can have different characteristics, you might have to experiment with different values for the SET FILE TYPE and SET FILE BLOCKSIZE commands before you find the one that works right for you.

ASCII This is the default file type. Incoming files are stored as standard VAX/VMS text files with variable length records and carriage return / line feed sequences implied between records (that is, with the CR carriage control record attribute). This is the format preferred by most utility programs under VAX/VMS. A fatal error occurs if any line is more than 4096 characters long. Note that incoming lines are only terminated by carriage return, line feed sequences. A carriage return that is not followed by a line feed or a line feed that is not preceded by a carriage return is not considered the end of a line, and is included within the body of a record.

BINARY Store received files with variable length records and no record attributes. Records are written using the current "blocksize". The last record may be short, with its record size correctly indicated. The default "blocksize" for binary files is 510, so that a record together with its two-byte length field exactly fill a 512-byte VMS disk block. Any file which is just a stream of bytes can be stored as a BINARY file, and recovered intact later. This is the preferred file type for use in archiving non-VMS files. The longest possible record is 32765 plus two bytes for the RMS length field.

BLOCK Store received files exactly as they come in, byte for byte, with no formatting or record length information. When sending files, send the file data literally, including record attributes (if any), and ignoring all RMS attributes. Using a file type of BLOCK has proven effective when transferring files between the same application on unlike systems, for example Lotus 1-2-3 spreadsheets between VMS and MS-DOS. When receiving a file in this mode, any unused portions of the last block are filled with zeros.

FIXED Store the file as a fixed-length-record binary file. Any file received is stored as fixed length records with no record attributes, using the current "blocksize" (i.e. record length, 512 by default). Fixed-length 512-byte records is the format used for binary files such as VAX/VMS "EXE" files and RSX-11M/M+"TSK" files. VMS BACKUP savesets are fixed-length-record files with a record-length ("blocksize") of 2048 or more. Since even the last record of the file is written with the whole record length (even if it is not filled), this format does not necessarily maintain the correct length of a file. It should normally only be used for fixed-length-record files coming from a VAX/VMS, PDP-11, or other system, when the fixed-length nature of the data must be preserved.

NAMING keyword

Determine the form of names to be sent with outgoing files and determine the translation performed on incoming file names. The choices are FULL, NORMAL_FORM and UNTRANSLATED.

FULL

Kermit-32 will send full file names (including device, directory, file name, file type and version number). When receiving a file, Kermit-32 will perform no translation of the file name (which must therefore be a legal VAX/VMS file specification).

NORMAL_FORM

Kermit-32 will send only the file name and file type. When receiving a file, Kermit-32 will convert the file specification received to contain only uppercase letters, digits, and at most one period. Any other characters will be translated to "X". There will be at most 39 characters before the period (if any), and at most 39 characters afterwards. This forces the file name to be a valid VAX/VMS file specification for VMS versions 4.0 and later. This is the default style of file naming.

UNTRANSLATED

Kermit-32 will send only the file name and file type. When receiving a file, Kermit-32 will not perform any conversions on the file specification, which therefore must be a legal VAX/VMS file specification. If you want to receive files with long names, use this option. To transfer files with VAX/VMS long names between two VMS 4.0-or-later systems, use this option on both sides.

SET HANDSHAKE

Syntax: SET HANDSHAKE ooo

Sets the half duplex line turnaround handshake character to the ASCII character whose octal value is ooo. Normally required for communication with half duplex systems like IBM mainframes in linemode.

SET IBM_MODE

Syntax: SET IBM_MODE ON or OFF

For communicating with IBM mainframes over half-duplex linemode connections. When IBM_MODE is set to ON, Kermit-32 will override the parity and local echo settings and use odd parity, local echo on, and also enable a handshake character of XON (control-Q, ASCII 021 octal). This feature allows Kermit-32 to exchange packets over half duplex connection with systems that wait for an XON before sending data.

The various features selected by this command can be overridden subsequently by SET PARITY, SET LOCAL_ECHO, and SET HANDSHAKE commands.

SET LINE

Syntax: SET LINE [terminal-name]

Specify the terminal name to use for file transfer or CONNECT; the terminal-name can be up to 255 characters long. If you issue this command using other than your job's controlling terminal, you will be running Kermit-32 in local mode, and you must log in to the remote system and run Kermit on that side in order to transfer a file. If you don't issue this command, Kermit-32 determines whether it is to run locally or remotely based on the default terminal line found when Kermit-32 is started. Kermit-32 uses a list of logical names to determine which terminal should be the default terminal line. The first of these names which translates to a terminal which is available (i.e., not allocated by some other process) is used. The logical names Kermit-32 tries are KER\$COMM, SYS\$INPUT, SYS\$\$OUTPUT, and SYS\$COMMAND. If none of these translate to an available terminal, Kermit-32 is running detached, and a terminal must be specified by the SET LINE command before any actions can be performed. If a terminal is found, Kermit-32 is running locally if this is a terminal other than the one controlling the job (i.e., different from SYS\$COMMAND), otherwise Kermit-32 is running remotely. You can also select the line directly in the CONNECT command; the command:

CONNECT TTA0

is equivalent to:

```
SET LINE TTA0  
CONNECT
```

If you type **SET LINE** with no argument, you will deassign any previous assigned line and revert to remote mode on your job's controlling terminal.

SET SERVER_TIMEOUT

Syntax: SET SERVER_TIMEOUT number

This specifies the number of seconds between timeouts during server command wait, 0 specifies that no timeouts should occur during server command wait. When a Kermit server times out, it sends a NAK packet. Some systems cannot clear piled-up NAKs from their input buffers; if you're using such a system to communicate with a Kermit-32 server, and you expect to be leaving the server idle for long periods of time, you should use this command to turn off server command-wait timeouts. This command is also useful when a server is connected to a modem that is waiting for a call to come in, in which case the server's NAKs could confuse the modem's autodialer.

SET TRANSMIT

Syntax: SET TRANSMIT DELAY integer, SET TRANSMIT ECHO ON/OFF

It is possible to set a few parameters associated with the raw **TRANSMIT** command that vary both what the user sees on the screen as well as the speed of the transmit.

SET TRANSMIT DELAY

This parameter is the amount of time to delay after each carriage return is transmitted. Valid delay values range between 0 (the default) and 9 tenths of a second. The format of the command is: **SET TRANSMIT DELAY d** Where d is a single decimal digit representing tenths of a second.

Some remote hosts may not be able to receive the characters as fast as Kermit-32 can send them. The **TRANSMIT DELAY** can be used to slow up the transfer by adding a slight delay after each line is sent.

The transfer also runs slower if the transmit echo is on, and the remote system is echoing the characters as it receives them. If the transmit delay is set to 9 tenths of a second, the remote system is echoing characters, the transmit echo is on, and the remote system still cannot keep up, then the connection should be made at a slower baud rate.

Conversely, the file transfer speed can be increased by: setting the delay to 0 and the echo off, stopping the remote system from echoing the characters it receives, and connecting at higher baud rates.

SET TRANSMIT ECHO

This command controls what the user sees on the screen during the file transfer. The format of the command is **SET TRANSMIT ECHO ON** or **OFF**. By default, the transmit echo is left off and the user sees the number of each line after it has been transmitted. With transmit echo on, the user sees whatever the remote system would normally echo back to him while he is typing in a file. Note that turning the echo on typically slows the file transfer down.

THE SHOW COMMAND

Syntax: **SHOW** [option]

The **SHOW** command displays various information:

ALL All parameters.

BLOCK_CHECK_TYPE The block check type being requested.

COMMUNICATIONS Parameters affecting the terminal line being used for communication.

DEBUGGING Debugging mode in effect, if any.

DELAY The number of seconds Kermit-32 will delay before starting a **SEND** or **RECEIVE** command when in remote mode.

ESCAPE The current escape character for the **CONNECT** processing.

FILE_PARAMETERS File blocksize, type, file naming, and incomplete file disposition.

INCOMPLETE_FILE_DISPOSITION The action to take when a transfer is aborted.

LINE Terminal line in use.

LOCAL_ECHO Whether characters should be echoed locally when **CONNECTed**.

PACKET For incoming and outbound packets.

PARITY The parity type in use.

RECEIVE For inbound packets.

RETRY The number of retries to be done on bad packets.

SEND For outbound packets.

TRANSMIT Parameters for **TRANSMIT** command.

VERSION The program version number of Kermit-32.

1.4.6 Program Management Commands

THE HELP COMMAND

Syntax: **HELP** [topic {subtopic}]

Typing **HELP** alone prints a brief summary of Kermit-32 and its commands. You can also type

HELP command

for any Kermit-32 command, e.g. "help send" or "help set parity" to get more detailed information about a specific command. The **HELP** feature depends on the Kermit-32 help file being correctly installed on your system.

THE EXIT AND QUIT COMMANDS

Syntax: **EXIT**

Exit from Kermit-32. You can also exit from the Kermit-32 when it is waiting for a command by typing a control-Z. When Kermit-32 is running remotely, two control-Y's will abort the transfer, bringing Kermit-32 back to command mode. The two control-Y's must be typed together; if a timeout occurs between them the first is ignored. When Kermit-32 is running locally, two control-Y's will stop Kermit-32 and return you to DCL. You will be able to CONTINUE if you do not perform any command which runs a program. However, after continuing, control-A, control-X and control-Z will no longer be accepted as commands.

QUIT is a synonym for **EXIT**.

THE LOG COMMAND

Syntax: LOG [option [filespec]]

Log the specified option to the specified file:

SESSION During CONNECT log all characters that appear on the screen to the specified file. During CONNECT, the session log can be temporarily turned off during the remote session by typing the escape character followed by Q (for Quit logging), and turned on again by typing the escape character followed by R (for Resume logging).

TRANSACTIONS During file transfer, log the progress of each file. Transaction logging is recommended for long or unattended file transfers, so that you don't have to watch the screen. The log may be inspected after the transfer is complete to see what files were transferred and what errors may have occurred.

DEBUGGING Log debugging info to the specified file. If no SET DEBUGGING command was previously issued, the file will be opened and no information written. If DEBUGGING is turned on (either via the SET DEBUGGING command or by typing control-D during a local transfer), the packet debugging information will be written to the file. Packet format is described in Kermit, A File Transfer Protocol, Frank da Cruz, Digital Press (1987).

Any log files are closed when you EXIT or QUIT from Kermit. You may explicitly close a log file and terminate logging by using the LOG command without a file specification.

THE STATUS COMMAND

Syntax: STATUS

The current status of Kermit-32 will be displayed. This includes the number of characters that have been sent and received from the remote Kermit. Also included is an estimate of the effective baud rate of the transfer. This number is not intended to be exact, but only an indication of what range of throughput has been provided.

1.5 Raw Upload and Download

THE TRANSMIT COMMAND

Syntax: TRANSMIT file-spec

The TRANSMIT command allows you to upload files "raw" to systems that don't have a Kermit program available. Note that there is no error checking or packets involved in this method of file transfer.

This command does a raw transmit of an ASCII file, one character at a time, with carriage returns (no line-feeds) at the end of each line. It is used with Kermit-32 in local mode. The user must first prepare the remote host to receive the file by starting an edit session in input mode. Then the user can escape back to Kermit-32 and issue the TRANSMIT command. After the transmit is finished, the user then CONNECTs back to the remote host again and ends the edit session.

During a file transmit, the following control characters can be used to affect the transfer in progress:

- | | |
|---------------|---|
| CTRL-C | Cancel the transmit |
| CTRL-X | Cancel the file currently being transmitted |
| CTRL-Z | Cancel the file group currently being transmitted |

See SET TRANSMIT for information about controlling echo and delays.

THE LOG SESSION COMMAND

Syntax: LOG SESSION file-spec

"Raw Download" is the term commonly used to describe the capture of a remote file on the local system, without any kind of error detection or correction. This allows you to obtain files from remote systems that do not have Kermit, but with the risk of loss or corruption of data.

Kermit-32 provides raw downloading via the LOG SESSION command during CONNECT to a remote system. The session log is described above. To use session logging to capture a file:

1. Run Kermit on the VAX/VMS system.
2. SET LINE to the terminal line through which you will be connected to the remote system.
3. Perform any required SET commands to condition Kermit for communication with the remote system.
4. CONNECT to the remote system and log in.
5. Condition your job on the remote system not to pause at the end of a screenful of text, and give whatever commands may be necessary to achieve a "clean" terminal listing -- for instance, disable messages from the system or other users.
6. Type the appropriate command to have the desired file displayed at the terminal, but do not type the terminating carriage return. On most systems, the command would be "type", on Unix it's "cat".
7. Escape back to Kermit-32 and give the LOG SESSION command with the file specification where you wish to store the data.
8. CONNECT back to the remote system and type a carriage return. The file will be displayed on your screen and recorded in the session log file.
9. Escape back to Kermit-32 and give the LOG SESSION command without a file specification to close the session log file.

The file you specified will contain everything that was typed on your screen. You will probably find that some editing necessary to remove extraneous prompts, messages, padding characters, or terminal escape sequences, or to fill in lost or garbled characters.

Use the TRANSMIT command for raw uploading.

1.6 Installation of Kermit-32

VMS Kermit-32 comes in 3 forms: Hex, Macro source, and Bliss source. Each can be used as the basis for installation.

Before beginning, make a special directory for VMS Kermit and read the files VMS*.* from the Kermit distribution tape into this directory. Columbia's 9-track Kermit tapes are written with blocksize 8192, which is 4 times larger than the default tape blocksize for VMS. You should mount these tapes on the VMS system with the following command:

```
MOUNT/BLOCK=8192/DENSITY=1600 MTA0: KERMIT
```

(or substitute some other tape drive name for MTA0:) Do not use the /FOREIGN switch. Once the tape is mounted, you can use normal VMS COPY commands to copy the files from the tape. For instance, if you have defined your Kermit directory to have logical name KER:, you can use the following command to copy the VMS Kermit files into this directory:

```
$ copy mta0:vms*.* ker:
```

You might also have received Kermit on a TK50 tape cartridge that contains a VMS BACKUP saveset, in which case do this to get the files off:

1. SET DEFAULT to the directory under which you want the various Kermit subdirectories created.
2. Physically mount the TK50 cartridge, and type "MOUNT \$TAPE1/FOREIGN".
3. Type "BACKUP/LOG \$TAPE1/SAVE [.*]".

Installation Procedure

If you are running a pre-4.0 version of VMS, ignore the following material and skip ahead to the section "Kermit-32 for Old VMS Versions".

At present, there is no VMSINSTAL "kit" for Kermit-32. However, there is a DCL procedure that will do most of the installation work for you. It is called VMSINS.COM. To run it, type:

```
$ @vmsins
```

It will ask you the question "Rebuild from sources? (YES or NO)". If you reply NO, then the Kermit task image will be decoded from the the VMSMIT.HEX file into KERMIT.EXE. If you reply YES, you will be given the choice of building the program from the Macro-32 sources (which are generated by the Bliss compiler from the original Bliss source code) or from the Bliss itself. All sites can build from Macro, but only those sites with Bliss compilers can build from the Bliss.

After building the KERMIT.EXE file, the VMSINS procedure copies it into SYS\$SYSTEM, and then builds and installs the Kermit-32 help file in the system-wide help library (SYS\$HELP:HELPLIB.HLB) so that users can get help for Kermit by typing "help kermit" at the DCL prompt, and it will also build and install SYS\$HELP:KERMIT.HLP so that the HELP command will work from within Kermit.

HEX, Macro, or Bliss?

The VMSMIT.HEX file is built from KERMIT.EXE under the oldest version of VMS that the developers have access to (for example VMS 4.5). If you are running that version of VMS or later, then you should reply NO to the "Rebuild from sources? (YES or NO)" question.

If you are running an older version of VMS than the one under which the Kermit that forms the basis of the hex file was linked, then you will not be able to run it on your VMS system, because of a runtime library conflict. In that case, you should reply YES to the question, and VMSINS will try to build the program from the Macro-32 assembly language source code using your system's MACRO command. This should build a working version of Kermit-32 on all VAX/VMS systems 4.0 or later.

The only reason for building from the Bliss source is if you have made changes to Kermit-32. It is recommended that you only work on the Bliss source, and not the Macro source. If you make changes to the macro source, there is no way to carry them forward to the Bliss code, which is the true source code for the program. If you do intend to make changes to the Bliss code, be sure to contact Columbia University's Kermit Distribution Center first to make sure you are working from the latest release and that nobody else has already done, or is working on, the same thing.

Kermit-32 for Old VMS Versions

If you are running a pre-4.0 version of VAX/VMS, then you will have to install a very old version (3.1) of Kermit-32, rather than the current version, until you upgrade your VMS version. To use version 3.1 of VMS Kermit:

1. Rename VMSMIT.HEX to VMSV33.HEX
2. Rename VMSV31.HEX to VMSMIT.HEX
3. Run the VMSINS procedure and reply NO to the "Rebuild from source" question.

Note that the help files which are installed apply to the current release, 3.3.126, rather than to version 3.3.

Defining a Kermit Command

You should define a system-wide symbol for Kermit as a "foreign command", for example in your SYS\$MANAGER:SYLOGIN.COM (system-wide login command) file, like this:

```
KERMIT := $SYS$SYSTEM:KERMIT.EXE
```

so that users can run Kermit just by typing its name.

Files

Kermit-32 is built from a number of BLISS-32 sources and one MACRO-32 source. In order to make it possible for sites without BLISS-32 to build, MACRO-32 sources generated by BLISS-32 are also included for all of the BLISS modules. The following files are distributed as part of Kermit-32:

VMSTT.BLI Common BLISS source for the terminal text output support.

VMSGLB.BLI Common BLISS source for the global storage for
VMSMSG.BLI.

VMSMSG.BLI Common BLISS source for the protocol handling module.

VMSCOM.REQ Common BLISS require file which defines various common parameters. This is required by VMSMSG.BLI. This file must be renamed to KERCOM.REQ.

VMSMIT.BWR "Beware File" for Kermit-32 (read it!).

VMSMIT.BLI BLISS-32 source for the command parser, and some basic support routines.

VMSFIL.BLI BLISS-32 source for the file I/O.

VMSTRM.BLI BLISS-32 source for the terminal processing. This handles the driving of the terminal line for the transfers and the connect command processing.

VMSSYS.BLI System interface routines for the Kermit generic command processing.

VMSGEN.MAR Macro-32 source file that contains the REMOTE command text that is given to VMS. Sites desiring to change what DCL commands are used to process the various generic server commands can make those changes in this source. This also contains the text of the help message returned in response to the server generic help command.

VMSERR.MSG MESSAGE source for error messages used by VAX/VMS Kermit.

VMSERR.REQ BLISS-32 require file which defines the error codes. This is REQUIRED by the BLISS-32 sources.

VMSMIT.MSS SCRIBE source file for VMSMIT.DOC (this document).

VMSMIT.RNH RUNOFF source for the help files for VAX/VMS Kermit. When this is run through RUNOFF with /VARIANT=SYSTEM, it produces a .HLP (VMSSYS.HLP) file suitable for inserting into the system help library (SYS\$HELP:HELPLIB.HLB) to provide a KERMIT topic for the system HELP command. When run through RUNOFF without the /VARIANT=SYSTEM, it produces a .HLP file (VMSUSR.HLP) to be stored on SYS\$HELP: for use by the Kermit HELP command.

VMSSYS.HLP RUNOFF output file for system wide Kermit HELP.

VMSUSR.HLP RUNOFF output file for Kermit's HELP command.

VMSREN.COM Command file to rename VMS*. * to KER*. *

VMSINS.COM Command file to build and install VAX/VMS Kermit.

VMSMIT.HEX A hexified version of .EXE file for VMS Kermit. This file can be dehexified using the supplied program. In the hexified form, the file should be transferable over any medium which handles normal text. This is the most reliable copy of the executable version of VMS Kermit.

VMSHEX.MAR Source for the hexification program. This is the program which was used to produce VMSMIT.HEX. It can also be used to produce hexified version of any (or at least almost any) Files-11 file. The dehexification program should then be able to reproduce a copy of the original file with the file parameters correctly set. Note that the format used for the hexified files is basically Intel hex format. There are some additional records used to store the record format, etc. Also, the file name as typed to the prompt from VMSHEX is stored in the hexified version of the file for use by the dehexification program. By doing this, it is possible to store more than one binary file with a single hexified file.

VMSDEH.MAR Source for the dehexification program.

VMSV31.* Version VMS Kermit, the last version that will run under release 3.x of VMS. Versions 3.2 and later require VMS release 4.0 or later.

VMSV3x.MEM Documentation on the changes between releases 3.1 and 3.1, and 3.2 and 3.3 of Kermit-32, and additional installation information.

OTHER INSTALLATION CONSIDERATIONS

As distributed, Kermit-32 should work on any VAX/VMS system (version 4.0 and later). Customization is possible with or without a BLISS-32 compiler. Default parameter values may be changed by changing the appropriate LITERALS in the BLISS-32 source for VMSMSG, or the actual values which are stored in the routine MSG_INIT in the MACRO-32 source for VMSMSG.

Sites can also easily change the commands which are used for processing the generic server functions (REMOTE commands when running as a server). The text which makes up these commands is in the file VMSGEN.MAR, along with the text of the REMOTE HELP message. This allows a site to make use of local programs for performing some of the commands (perhaps using FINGER to perform the WHO command, etc.).

If you want to allow your users to assign external terminal lines for connecting to remote systems from the VAX, e.g. by dialing out, you will have to configure those lines to allow the desired access. Otherwise, users will get a message like "No privilege for attempted operation" when they do a SET LINE command. Sample commands for terminal TXA0: might include:

```
$ SET PROTECTION=(W:R) TXA0:/DEVICE
```

or

```
$ SET PROTECTION=(W:RWLP)/DEVICE/OWNER=[1,4] TXA0:
```

or

```
$ SET ACL /OBJECT=DEVICE /ACL=(IDENTIFIER=INTERACTIVE,-  
  OPTIONS=NONE, ACCESS=READ+WRITE) TXA0:
```

Consult your VAX/VMS system manager's manual for the ramifications (especially on security) of each of these commands.