gamma11

GAMMA-11
System Reference

Order No. AA-2186B-TC

June 1978

This document describes the GAMMA-11 patient files, save area files, and playback files. This document also describes the BASIC and FORTRAN routines that access these files for user-written programs. See also the *GAMMA-11 Operator's Guide* (AA-2185B-TC).

SUPERSESSION/UPDATE INFORMATION: This document completely supersedes the document of the same name, Order No. DEC-11-MGRMA-A-D, published August 1976.

OPERATING SYSTEM AND VERSION: RT-11 V03B

SOFTWARE VERSION: GAMMA-11 V2C

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

GAMMA-11
System Reference

Order No. AA-2186B-TC

digital equipment corporation · maynard, massachusetts

First Printing August 1976
Revision June 1978

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10

CONTENTS

			Page
CHAPTER	1	BUILDING A GAMMA-11 SYSTEM DISK	1-1
	1.1	INTRODUCTION	1-1
	1.2	COPYING THE GAMMA-11 DISTRIBUTION MEDIA	1-2
	1.2.1	Copying a GAMMA-11 Distribution Disk	1-2
	1.2.2	Copying the GAMMA-11 Distribution Magnetic Tape	1-3
	1.3	CONFIGURING A GAMMA-11 SYSTEM DISK	1-4
	1.4	BACKING UP A GAMMA-11 SYSTEM DISK	1-8
	1.4.1	Disk-to-Disk Copy	1-8
	1.4.2	Disk-to-Magnetic-Tape	1-9
	1.4.3	Magnetic-Tape-to-Disk Copy	1-9
	1.5	UNSUPPORTED PROGRAMS ON THE GAMMA-11 V2C MEDIA	1-10
CHAPTER	2	PROGRAMMING SUPPORT INFORMATION	2-1
	2.1	GAMMA-11 PATIENT FILES	2-1
	2.1.1	Dynamic Studies	2-2
	2.1.2	Single Static Studies	2-3
	2.1.3	Multiple Static Studies	2-4
	2.1.4	List Mode Studies	2-6
	2.2	Z-COUNT AREA	2-6
	2.3	COMMENT AREA	2-6
	2.4	LAYOUT AND DESCRIPTION OF THE ADMINISTRATIVE DATA BLOCK	2-7
	2.4.1	The Administrative Data Block Common Section	2-7
	2.4.2	The Administrative Data Block Individual Study Section	2-7
	2.5	PATIENT FILENAMES	2-10
	2.6	SAVE AREAS	2-11
	2.6.1	Save Area Descriptor Block	2-11
	2.6.2	Saving Matrix Data	2-15
	2.6.3	Saving Dynamic Curves	2-17
	2.7	INTERNAL GAMMA-11 FILES	2-17
	2.8	GAMMA-11 MACRO AND PLAYBACK FILES	2-18
	2.8.1	Playback Files	2-18
CHAPTER	3	BASIC AND FORTRAN SUPPORT	3-1
	3.1	BASIC AND FORTRAN SUPPORT FOR GAMMA-11 F/B	3-1
	3.1.1	Support Routine Notation	3-1
	3.1.2	Patient Data File Subroutines	3-2
	3.1.3	Save Area Subroutines	3-6
	3.1.4	General Purpose Support Subroutines for BASIC and FORTRAN	3-10

			Page
	3.1.5	Linking FORTRAN Subroutines with a User Program	3-11
	3.1.6	BASIC And FORTRAN Error Messages	3-11
	3.1.7	BASIC and FORTRAN Examples	3-15
	3.2	SUPPLEMENTAL FORTRAN SUPPORT	3-19
	3.2.1	FORTRAN and GAMMA-11 Variables	3-19
	3.2.2	Arrays	3-19
	3.2.3	Functions	3-20
	3.2.4	Subroutines	3-20
	3.2.5	Linking Supplemental FORTRAN Subroutines With A User Program	3-22
	3.2.6	FORTRAN Example	3-22
CHAPTER 4		ASSEMBLING AND LINKING GAMMA-11	4-1
	4.1	ASSEMBLING GAMMA-11 USING INDIRECT COMMAND FILES	4-1
	4.2	LINKING GAMMA-11 USING INDIRECT COMMAND FILES	4-5
APPENDIX A		BASIC/RT-11 LANGUAGE SUMMARY	A-1
	A.1	BASIC/RT-11 STATEMENTS	A-1
	A.2	SUMMARY OF BASIC/RT-11 FUNCTIONS	A-5
		Arithmetic Functions	A-5
		String Functions	A-6
		System Functions	A-7
	A.3	SUMMARY OF BASIC/RT-11 COMMANDS	A-8
		Key Commands	A-9
	A.4	BASIC/RT-11 ERROR MESSAGES	A-10
		Function Error Messages	A-17
APPENDIX B		FORTRAN/RT-11 LANGUAGE SUMMARY	B-1
	B.1	EXPRESSION OPERATORS	B-1
	B.2	STATEMENTS	B-2
	B.3	FORTRAN LIBRARY FUNCTIONS	B-11
APPENDIX C		CAMERA ORIENTATION	C-1
	C.1	TRANSFORMATION OPERATORS	C-1
	C.2	CONVERSION TO OTHER CAMERAS	C-3
APPENDIX D		USING A NEW DISK	D-1
	D.1	FORMATTING A NEW RK05 DISK ON AN 11/34	D-1
	D.2	INITIALIZING AN RK05 RT-11 DISK DIRECTORY	D-2
	D.3	INITIALIZING AN RK06 RT-11 DISK DIRECTORY	D-3
INDEX			Index-1

CONTENTS (CONT.)

Page

FIGURES

FIGURE	2-1	Dynamic Study Data File	2-2
	2-2	Single Static Study Data File	2-3
	2-3	Multiple Static Study Data File	2-5
	2-4	List Mode Study Data File	2-6
	2-5	Matrix Save Area	2-15
	2-6	ROI Map and Cell Map	2-16
	2-7	Word in ROI Map	2-16
	2-8	Dynamic Curve Save Area	2-17

TABLES

TABLE	2-1	Administrative Block	2-8
	2-2	Save Area Descriptor Block	2-11
	3-1	Administrative Data Block	3-4
	3-2	Save Area Descriptor Block	3-7

CHAPTER 1

BUILDING A GAMMA-11 SYSTEM DISK

1.1 INTRODUCTION

This chapter describes the procedures that are required in order to build a GAMMA-11 system disk. GAMMA-11 is distributed on the following media.

- RK05 disks
- RK06 disks
- RL01 disks
- Magnetic tape

The disks are complete, runnable system disks that contain all the GAMMA-11 software and an executable subset of the RT-11 V3B software. A special version of BASIC with GAMMA-11 support subroutines and binary files providing FORTRAN callable GAMMA-11 support subroutines are on the disks. The magnetic tape is a bootable tape that can generate an RK05, RK06, RK07, or RL01 system disk.

NOTE

> Although the GAMMA-11 distribution disk is a runnable system disk, it should not be used as such. You should copy the distribution disk, back it up, and then store it in a safe place. In your daily activities, you should use only the copies that you make of the sysgened disk.

The general procedure for building a GAMMA-11 system disk is as follows:

1. Copy the GAMMA-11 distribution disk or magnetic tape to a scratch disk. (See Section 1.2.)
2. Run the SYSGEN program, which creates a configuration procedure that will tailor the system to your needs. (See Section 1.3.)
3. (Optional) Add RT-11 FORTRAN to your system disk.
4. Backup your new system disk on another disk or magnetic tape. (See Section 1.4.)

The following sections give step by step instructions for performing the above procedures.

The generic terms 'disk', 'disk pack', 'disk drive', etc. refer to either the RK05, RK06, RK07 or RL01 disks. The general build procedures are the same for all of these disks, except where specifically noted.

1.2 COPYING THE GAMMA-11 DISTRIBUTION MEDIA

GAMMA-11 is distributed on RK05, RK06, and RL01 disks and 9-track magnetic tape. This section gives the instructions for copying the distribution disks or magnetic tape to a scratch disk. You will use this copy of the GAMMA-11 distribution when building your system.

The build procedures for the different disk types are the same, the only difference being the 2-character device mnemonic used to identify the various disks. In the sections below, you must replace any occurrence of 'xx' with the 2-character device mnemonic that identifies your type of system disk. The mnemonics are:

RK for the RK05 disk
DM for the RK06 or RK07 disk
DL for the RL01 disk

1.2.1 Copying a GAMMA-11 Distribution Disk

Step 1: Mount the GAMMA-11 distribution disk in unit 0 of the disk drive. Write lock the disk by pressing the 'WTPROT' or 'WRITE PROT' switch. The corresponding light should come on, indicating that the disk is indeed write protected. Mount a formatted scratch disk in drive 1; this disk must not be write protected.

Step 2: Bootstrap the distribution disk. See Chapter 3 of the GAMMA-11 Operator's Guide for bootstrap instructions. When the disk has been bootstrapped, the following will be printed.

```
RT-11FB V03B-nn

.TYPE WARN.TXT

                    WARNING

You have just booted your MASTER copy of GAMMA-11
F/B V02C.  Please copy this disk according to the
instructions in Section 1.2.1 of the GAMMA-11 System
Reference manual.
```

Step 3: To copy the master disk (in drive 0) to the scratch disk (in drive 1), type:

```
@MSTCOP
```

The initialization and copy procedure will take from 1 to 5 minutes depending on the disk type. During this time, the system will print a few lines of commands on the DECwriter.

When the copying has been completed, the system will print the following message.

```
COPY COMPLETE
```

Step 4: The disk in drive 1 is now a copy of the disk in drive 0. Remove the distribution disk from drive 0 and store it in a safe place. Remove the new system disk from drive 1 and mount it on drive 0. Proceed to Section 1.3 to configure your new system disk.

1.2.2 Copying the GAMMA-11 Distribution Magnetic Tape

Step 1: Mount the GAMMA-11 distribution magnetic tape on the magnetic tape unit. Mount a formatted scratch disk in drive 0. This disk must not be write protected.

Step 2: Bootstrap the magnetic tape. If your processor has a hardware bootstrap, such as a PDP-11/34, boot the processor and type:

```
MT
```

Otherwise, refer to Appendix C of the RT-11 System Generation manual for instructions on how to bootstrap a TM11 magnetic tape unit.

When the tape has been successfully booted the system prints the following message.

```
MSBOOT V01-nn
*
```

Step 3: Start the TM11 magnetic tape build program by typing the following line at the asterisk.

```
MDUP.MT
```

The program will respond with an '*'.

Step 4: Initialize and scan the scratch disk for bad blocks by typing:

```
xx0:/Z/B
```

where xx is RK, DM, or DL.

The scan will take a few minutes. When the scan is complete the system will print '*'.

Step 5: Copy a minimal RT-11 system to the disk by typing:

```
xx0:A=MT:
```

where xx is RK, DM, or DL.

Then the disk will be booted. When it is booted, the system will print the following message.

```
RT-11SJ V03B-nn
?KMON-F-Command file not found
```

Step 6: Copy the GAMMA-11 magnetic tape copy file from the magnetic tape to the disk by typing:

```
COPY MT:MTTOxx MTCOPY
```

where xx is RK, DM or DL.

The system response is:

```
 Files copied
MT:MTTOxx.COM to xx:MTCOPY.COM
```

Step 7: Proceed with the magnetic tape copying procedure by typing:

```
@MTCOPY
```

The copy procedure will take a few minutes. The system will print a few lines of commands during the copying. When the copy is completed the system will reboot itself.

Proceed with step 2 of the next section.

1.3 CONFIGURING A GAMMA-11 SYSTEM DISK

After you have copied your GAMMA-11 distribution medium, you must configure it for your specific GAMMA-11 system. You do this using a program called SYSGEN, which asks you a series of questions concerning your system and then generates an RT-11 indirect command file and BATCH file that do the actual configuring of your system disk.

Step 1: Mount a disk copy of the GAMMA-11 distribution medium in drive 0 and boot it. The following message will be printed:

```
RT-11FB V03B-nn

.TYPE GAMCOP.TXT

This is a copy of the GAMMA-11 F/B V02C distribution
media.    You    should    follow    the    configuration
instructions in Section 1.3 of the GAMMA-11 System
Reference manual.
```

Step 2: To configure your disk type:

```
R SYSGEN
```

An explanation of the system configuration will be printed. You will be asked a series of questions concerning your system. Each question is preceded by a short explanation that will aid you in answering the question. The example below was used to generate a standard RK05 system.

```
.R SYSGEN


GAMMA-11 V02C SYSTEM CONFIGURATION


This procedure will configure your GAMMA-11 V2C
disk to your hardware configuration.  You will
be asked a series of questions.  All questions
except the first two are to be answered with


Y for yes
N for no


followed by a carriage return.  If you type
just a carriage return, 'yes' will be used.


After answering all the questions, you will be
given further instructions.  If you make a
mistake and wish to restart, type a CTRL/C, and


R SYSGEN


------------------------------------------------


GAMMA-11 can use any of the following disks as
system devices.  (RK05 is the default.)


RK05    RK06    RK07    RL01


WHAT IS YOUR SYSTEM DISK? RK05


The GAMMA-11 disk must be loaded in an RK05 drive, and
the disk unit must be up to speed and not write protected
before you answer the next question.


IN WHICH DISK UNIT DID YOU LOAD THE GAMMA-11 DISK? 0


You have a choice between a foreground/background
operating system, or a single job operating system.
The foreground/background operating system will
allow you to simultaneously acquire and analyze data.
Most users will use this feature, the only users
who can not use it are those with no foreground
terminal, or less than 28K words of memory.


DO YOU WANT A FORGROUND/BACKGROUND SYSTEM? Y
```

If your PDP-11 has the Extended Instruction Set (EIS) you can use the EIS version of BASIC. EIS is standard on a PDP-11/34.

IS YOUR SYSTEM A PDP-11/34 OR DOES IT HAVE EIS? Y

The RT-11 Monitors will have to be patched if your line frequency is 50 Hz instead of 60 Hz.

IS YOUR LINE FREQUENCY 60 HZ? Y

The standard display for a GAMMA-11 system is the VSV01 color display. Some users may have the older VT01 storage scope display.

DOES YOUR SYSTEM HAVE A VSV01 COLOR DISPLAY? Y

The Gate Synchronized Acquisition (GSA) programs require an external gating signal. If your system has the new NCV11-A gamma camera interface, the gate is always interfaced through it. If you have the NC11-A interface, the gate is interfaced via the AR-11.

DOES YOUR SYSTEM HAVE THE NEW NCV11 GAMMA CAMERA INTERFACE? N

If your gate signal is a TTL low-to-high signal, it is interfaced thru AR-11 ADC channel 3.
If your gate signal is a TTL high-to-low signal, it is interfaced thru the AR-11 'EXT A/D ST'.

WILL YOU INPUT THE GATE SIGNAL THRU AR-11 ADC CHANNEL 3? Y

The data acquisition programs can store the date in either the U.S. date format, month/day/year, or the European date format, day-month-year.

DO YOU WISH TO USE THE U.S. DATE FORMAT, MONTH-DAY-YEAR? Y

The magtape backup procedure can create bootable or non-bootable magtapes. Bootable magtapes require extra RT-11 programs. Only those users with a magtape drive and only one disk drive need the bootable magtape backup procedure. All others should delete this option.

DO YOU WISH TO DELETE THE BOOTABLE MAGTAPE OPTION? Y

The following system will be configured

A F/B RK05 system with
  VSV01 color display
  EIS support
  60 Hz line frequency
  U. S. date format (month/day/year)
  GSA input AR-11 ADC Channel 3

IS THIS CORRECT? Y

The actual configuration will take a few minutes.
When it is finished, the newly configured disk
will be booted. This disk should be backed-up
on another disk or mastape. Read section 1.3
of the GAMMA-11 SYSTEM REFERENCE MANUAL AA-2186B-TC.

You are now ready to run the actual configuration
files. The disk loaded in RK05 drive 0 will
be the disk that will be configured.
When the RT-11 MONITOR prints a dot (.), type

@RK0:GAMCNF (followed by a return)

When BATCH prints an asteric (*), type

GAMBAT (followed by a return)

```
.@RK0:GAMCNF
.LOAD BA
.ASSIGN RK0 LST
.ASSIGN RK0 LOG
.ASSIGN RK0 DK
.R BATCH
*GAMBAT
```

Step 3: Invoke the configuration procedure by typing:

```
@xx0:GAMCNF
```

where xx is RK, DM, or DL.

When the following is printed:

```
.R BATCH
*
```

type:

```
GAMBAT
```

The configuration will take a few minutes. The system will reboot itself when the configuration is complete, and prints the following:

```
RT-11FB V03B-nn (or RT-11SJ V03B-nn)
.RENAME/NOLOG START%.TMP *.COM
```

System configuration is now complete.

Users who have purchased FORTRAN Version 2 should install FORTRAN on their new system disk according to the instructions in the RT-11/F4 Installation Guide, and then proceed with the back up procedure.

1.4 BACKING UP A GAMMA-11 SYSTEM DISK

You should prepare a back-up copy of your new system disk on another disk or magnetic tape. If anything should happen to your system disk, the back-up copy can be used to quickly create a new system disk.

The configuration procedure leaves two RT-11 indirect command files on your disk which you can use to quickly and easily back up your disks. BACKUP is used for disk-to-disk copying, and MTBACK is used for disk-to-magnetic tape copy. The following sections describe the simple procedures required to back up your GAMMA-11 system disk.

1.4.1 Disk-to-Disk Copy

To back up your system on another disk, do the following:

Step 1: If your system disk is not running in drive 0, mount it in drive 0 and boot it. Write protect the system disk.

Step 2: Mount a scratch disk in drive 1. Do not write protect the scratch disk.

Step 3: Assign the scratch disk to device 'COP' by typing

```
ASSIGN xx1 COP
```

where xx is RK, DM, or DL.

Step 4: Initiate the copy by typing

```
@BACKUP
```

BACKUP will initialize the disk in drive 1 and scan it for bad blocks. Then all files will be copied from the system disk, and finally the bootstrap program will be copied. When this is finished (5-10 minutes), the disk in drive 1 will be an exact copy of the disk in drive 0. Simply repeat this section whenever a new copy of your system disk is needed.

1.4.2 Disk-to-Magnetic-Tape

To copy your system disk to magnetic tape, do the following:

Step 1: If it is not running, mount your system disk in drive 0 and boot it, write protected.

Step 2: Mount a scratch magnetic tape and place it on-line.

Step 3: Initiate the copy by typing:

```
MTBACK
```

MTBACK will initialize the magnetic tape and copy all files to it. If you did not request that the bootable magnetic tape option be deleted during system configuration, the magnetic tape will be a bootable magnetic tape.

1.4.3 Magnetic-Tape-to-Disk Copy

There are two methods by which a system disk can be generated from a magnetic tape back-up. Method 1 is the simplest; it requires a dual disk system and a running GAMMA-11 system disk. Method 2 is more complicated and is necessary only if your system has only one disk drive, or if no GAMMA-11 system disk is available.

METHOD 1 (non-bootable magnetic tape)

Step 1: Mount any GAMMA-11 system disk in drive 0 and boot it.

Step 2: Mount the magnetic tape and place it on-line.

Step 3: Mount a formatted scratch disk in drive 1.

Step 4: Assign the magnetic tape to device 'DK' and the scratch disk to device 'COP' by typing:

```
ASSIGN MT DK
ASSIGN xx1 COP
```

where xx is RK, DM, or DL.

Step 5: Initiate the copying by typing:

```
@BACKUP
```

BACKUP will initialize the disk in drive 1 and scan it for bad blocks. Then all files will be copied from the magnetic tape, and finally the bootstrap program will be copied. The entire process will take 5-10 minutes.

METHOD 2 (bootable magnetic tape)

Step 1: Mount the magnetic tape on the tape drive and mount a scratch disk in drive 0.

Step 2: Follow steps 2, 3, 4, 5, 7 of Section 1.2.2, copying a GAMMA-11 distribution tape.

Do not configure the disk, since it is a copy of a configured disk.

1.5 UNSUPPORTED PROGRAMS ON THE GAMMA-11 V2C MEDIA

Four unsupported programs are distributed with GAMMA-11 F/B V02C. These programs are furnished as a convenience to the user. These programs are NOT supported by DIGITAL.

VTECO and STECO are modified versions of the unsupported RT-11 text editor TECO. VTECO uses the VSV01 color display, and STECO uses the VT01 storage scope display in the same manner as TECO uses the VT11 graphics processor.

TTY1 and TTY0 allow the user to switch control of the RT-11 background to and from the foreground VT52 terminal.

The file CLASSC.TXT on the distribution media contains more information concerning these four programs. To print this file, mount the distribution medium on a spare drive, write protected and type:

```
TYPE dev:CLASSC.TXT
```

To copy any of these programs, type:

```
COPY dev:name DK:
```

where dev: is the device and unit of the distribution medium; e.g., DK1:, MT:, etc. The 'name' is the name of the program to be transferred.

CHAPTER 2

PROGRAMMING SUPPORT INFORMATION

2.1 GAMMA-11 PATIENT FILES

A GAMMA-11 file is an RT-11 file produced by the GAMMA-11 programs. Consequently a GAMMA-11 file can be treated like any RT-11 file and can be read by BASIC, FORTRAN, or assembly language programs.

A different file structure represents each of the four GAMMA-11 patient study types. The four types of studies are: dynamic, single static, multiple static, and list mode.

A description of each of the four file types follows. Along with each description is a diagram of the file layout. The pointers labeled in each diagram are not explained in the general file descriptions. Instead they are explained in Section 2.4, and the whole file is laid out in Tables 2-1 and 2-2.

2.1.1 Dynamic Studies

A dynamic study consists of up to 512 frames collected at specified rates over designated periods of time, comparable to a sequence of motion picture frames. The specified rate of acquisition may be changed up to 12 times during the acquisition. Thus, a dynamic study consists of between one and thirteen groups of frames. Between groups only the frame rate, number of frames, and the matrix size may be varied.

Figure 2-1 shows the structure of a dynamic study file. The file consists of three to six blocks of general file and specific patient information followed by the gamma camera data.

The initial block (block 0) of a dynamic study file is called the administrative data block. This block is filled by the collection procedure program and contains all information entered by the user at acquisition set up time. It contains the patient name and number, organ under study, types of tests, dosage, and other statistical information together with pointers to various other blocks that make up the data file.

The z-count area follows the administrative data block and consists of one to four disk blocks, depending on the number of groups within the dynamic study. The z-count area contains statistics on the number of events that occurred during the study (see Section 2.2).

Following the z-count area is the comment block. The comment block holds the user's comments about the study (see Section 2.3).

The rest of the study is composed of the matrix data.

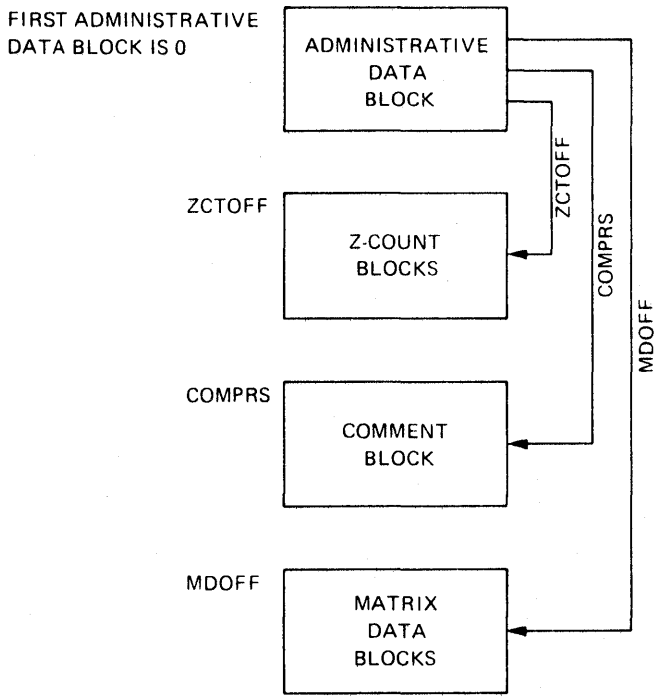


Figure 2-1 Dynamic Study Data File

2.1.2 Single Static Studies

A single static study is structured similarly to a dynamic study. However, in a single static study, the z-count area (see Section 2.2) is contained within the administrative block and only one frame of data follows the comment block (see Figure 2-2).

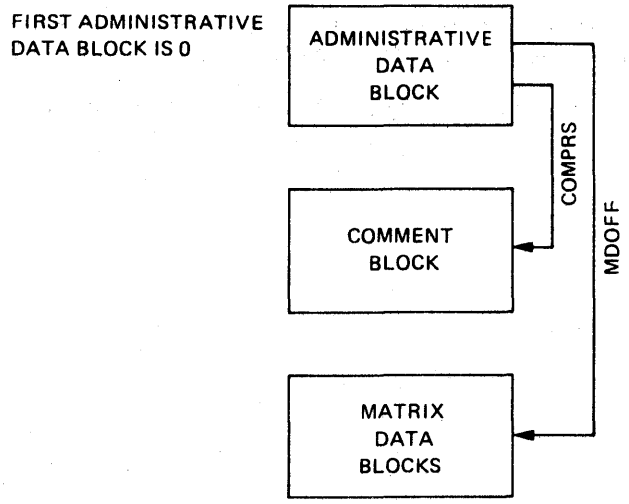


Figure 2-2 Single Static Study Data File

2.1.3 Multiple Static Studies

A multiple static study file is a number of single static study files linked together. In a multiple static study,

- each data frame is preceded by an administrative block, but only the first administrative block is followed by a comment block,
- each administrative block is linked to the administrative blocks that precede it and follow it,
- each data frame has a link to the comment block.

See Figure 2-3.

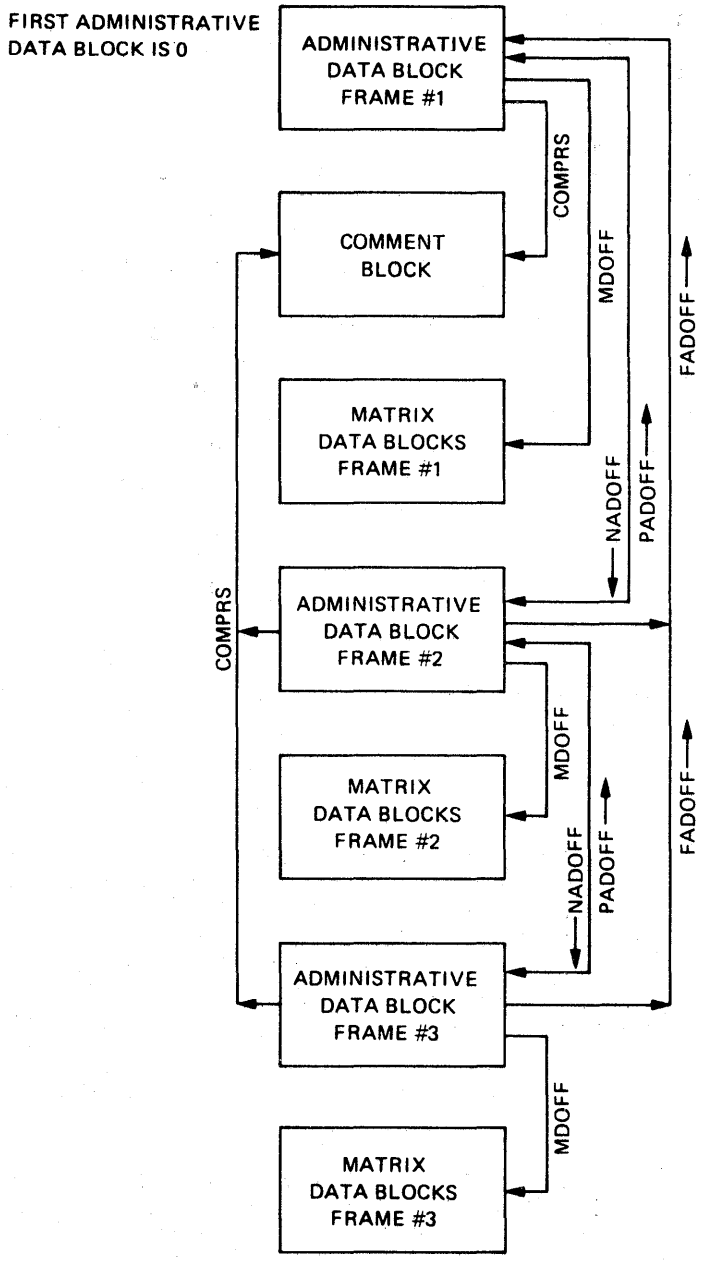


Figure 2-3 Multiple Static Study Data File

2.1.4 List Mode Studies

A list mode study is structured like a static study except that the data following the comment block is raw list data rather than a matrix. See Figure 2-4.

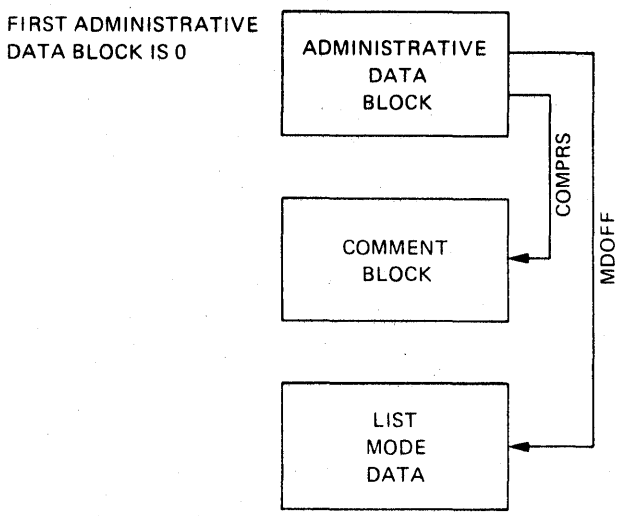


Figure 2-4 List Mode Study Data File

2.2 Z-COUNT AREA

The z-count area is contained in the administrative block for static and list studies and follows the administrative block for dynamic studies. The z-count area stores the number of events present on each study frame as a 32-bit unsigned integer composed of two 16-bit words. The first word is the high order 16 bits, and the second word is the low order 16 bits. In a dynamic study, the z-count area is one to four blocks of disk space.

2.3 COMMENT AREA

The comment block is available for the user's comments. The comment block consists of ten lines of ASCII text with up to 51 characters per line. The first character of each line is a non-printing character which is either an octal 0 or an octal 1. If the first character is an octal 1, the rest of the line contains up to 49 characters of valid ASCII text which is terminated by an octal 200. The first line with a 0 as the first character indicates the end of the comments.

2.4 LAYOUT AND DESCRIPTION OF THE ADMINISTRATIVE DATA BLOCK

The administrative data block contains all the information needed to reference the data in the study file. It is divided into two sections; the first section contains those variables and pointers common to all types of studies, and the second section contains those variables and pointers specific to each type of study.

Table 2-1 shows a complete layout of the administrative data block. The decimal and octal positions of each variable are given along with the variable type, its name, and its description.

2.4.1 The Administrative Data Block Common Section

The first section of the administrative data block, which occupies the area from octal address 0 to octal address 332, is common to all the study types. This section includes all offset pointers and the information pertinent to the patient such as the patient name, number, birth date, and doctor.

The offset pointers are the links from the administrative data block to the other data blocks. The offset pointers are:

ZCTOFF	points to the z-count block (dynamic study only)
COMPRS	points to the comment block
MDOFF	points to the data block

Those pointers that are specific to multiple static studies are:

PADOFF	points to the previous administrative block
FADOFF	points to the first administrative block
NADOFF	points to the next administrative block

2.4.2 The Administrative Data Block Individual Study Section

The second section of the administrative data block, which occupies octal positions 346 to 776, consists of collection parameters, number of frames, number of groups, types of matrices, and general administrative information pertaining to the immediate study.

The second section is an overlay area and is used for one type of study at a time. Since static, dynamic, and list studies cannot be combined in the same file, only the information applicable to the specific study type is used in the overlay area.

In the following table, Table 2-1, the variable types are abbreviated. ASC represents ASCII, INT represents integer, DPI represents double precision integer, and SPE represents special.

Table 2-1
Administrative Block

Decimal	Octal	Type	Name	Description
1	1	ASC	PATNAM	Patient name
24	30	ASC	PATNUM	Patient number
40	50	ASC	CAMID	Camera number (0-3) (NCV11 only)
43	53	ASC	ATIME	Acquisition time (supplied by program)
53	65	ASC	ADATE	Acquisition date (supplied by program)
63	77	ASC	BIRTHD	Patient birth date
73	111	ASC	DOC	Doctor's name
96	140	ASC	ORGAN	Organ being studied
110	156	ASC	VIEW	View of picture
120	170	ASC	CMTRT	Collimator type
127	177	ASC	AQMODE	Acquisition mode:1=special,2=normal
130	202	ASC	ISOTOP	Isotope being used
144	220	ASC	DOSE	Dosage
158	236	ASC	ISO2	2nd isotope being used
172	254	ASC	DOSE2	2nd dosage (dual isotope study)
186	272	ASC	ISMODE	1=single isotope, 2=dual isotope
189	275	SPE	COLTYP	Collection type
		BYTE	< 0	List mode
			= 0	Dynamic study
			> 0	Static study
192	300	INT	COMPRS	Offset to comment block
194	302	INT	TOTBLK	Total number of blocks in study
198	306	INT	FADOFF	Offset to first admin block (multiple static)
202	312	INT	DATTYP	Data type:0=patient data,1=flood
205	315	ASC	ORIENT	Orientation switch
208	320	ASC	POSSWT	Position (rotation) switch
212	324	INT	MDOFF	Offset to matrix data
214	326	INT	PADOFF	Static: offset to previous admin block (multiple static)
214	326	INT	ZCTOFF	Dynamic: offset to z count block
216	330	INT	NADOFF	Offset to next admin block (multiple static)
219	333	ASC	AMACRO	Auto analysis macro name
227	343	ASC	AUTO	Auto analysis switch (Y or N)
			STATIC AND LIST COLLECTION PARAMETERS	
230	346	DPI	ZCOUNT	Z count, the number of events
234	352	DPI	OVFTIM	Time of overflow clock counter
238	356	ASC	ENDFRA	Method of ending study:1=time, 2=counts
247	367	ASC	MINUTE	The number of minutes in the study
252	374	ASC	SECOND	The number of seconds in the study
257	401	ASC	PSCNT	The number of preset counts chosen
			STATIC MODE COLLECTION PARAMETERS	
241	361	ASC	SMTXSZ	The type of matrix (1,2,3,4, or 5)

Table 2-1 (Cont.)
Administrative Block

Decimal	Octal	Type	Name	Description
244	364	ASC	SMTXCS	Close on overflow: <SPACE>=do not close,<T>=close
268	414	ASC	MSFRM	The number of frames
272	420	INT	CFRM	Current frame number
			LIST MODE COLLECTION PARAMETERS	
241	361	ASC	LDBLCK	The number of disk blocks of data
268	414	ASC	LDBPC	Method of closing:1=by counts, 2=by number of blocks
271	417	ASC	LDELST	Method of starting:Y=delayed start,N=immediate start
274	422	ASC	LDELRT	Count rate for delayed start (maximum of 20,000)
281	431	ASC	LGSA	If 'Y', study is a gated list mode
			DYNAMIC MODE COLLECTION PARAMETERS	
230	346	INT	FRAMEN	Total number of frames
232	350	INT	GROUPN	Total number of groups
234	352	ASC	GRP1	The number of frames in group 1
238	356	ASC		The type of matrix(1,2,3 or 4)
240	360	ASC		The type of close
242	362	ASC		Frame rate: X frames
246	366	ASC		Per Y seconds (milliseconds for gated studies)
251	373	ASC	GRP2	Group 2 (same 5 parameters as group 1)
268	414	ASC	GRP3	Group 3
			.	
			.	
			.	
438	666	ASC	GRP13	Group 13
			GATE SYNCHRONIZED ACQUISITION (GSA) COLLECTION PARAMETERS (Group #1 is set up as a dynamic study)	
251	373	ASC	GSAMTX	GSA matrix type (1 or 2)
254	376	ASC	GSAFRM	Number of frames
259	403	ASC	GSADUR	Frame duration in msec
265	411	ASC	GSATOL	Tolerance in msec
271	417	ASC	GSAEFM	End Frame (1, 2, or 3) 1 = Time 2 = Counts 3 = Cycles
274	422	ASC	GSAPSC	Preset Counts
285	435	ASC	GSACYC	Preset Cycles
291	443	ASC	GSAMIN	Preset Minutes

Table 2-1 (Cont.)
Administrative Block

Decimal	Octal	Type	Name	Description
296	450	ASC	GSASEC	Preset Seconds
302	456	INT	BINSEC	Collection time in seconds
304	460	INT	BINCYC	# cycles collected (accepted plus rejected)
306	462	DPI	BINZLO BINZHI	Total Counts Collected
310	466	INT	BINDUR	Frame duration in msec
312	470	INT	BINTOL	Tolerance in msec
314	472	INT	BINBAD	# rejected cycles

2.5 PATIENT FILENAMES

GAMMA-11 identifies patient studies via an indexed line which contains up to 62 characters. For example,

1 JOHN DOE, 370180, LIVER, D, 6/3/74

is the way John Doe's file would appear on the GAMMA-11 patient study index. Internally, however, the patient files are referenced using RT-11 filenames. The RT-11 patient filenames have the form:

aaaaaa.Xnn — where aaaaaa are the first six characters of the name field of the patient's study. The nn is a number between 00 and 99. This formula creates a unique filename even when the name portion (aaaaaa) of the file is duplicated. The system assigns the number at the time of file creation. The numbers are assigned in the order of acquisition for each patient with the same name. For example, JOHNDO.X00 and JOHNDO.X01 are two filenames for two John Doe's (or two studies on the same John Doe).

These filenames are created automatically by GAMMA-11 at data acquisition setup time.

To reference GAMMA-11 data files when using BASIC or FORTRAN, you must know the RT-11 filename for that data file. To obtain the RT-11 filenames for all patients, type

F (RET)

when the patient index is displayed during the data analysis program. The RT-11 filenames will be displayed at the end of each index line instead of the date. For example,

1 JOHN DOE, 370180, LIVER, D, JOHNDO.X00

is displayed for the first John Doe.

2.6 SAVE AREAS

Save areas are disk files that are reserved for the user to store single matrices or dynamic curves. Save areas 0 through 9 permanently exist on the disk.

Save area 0 is used by the data analysis program for temporary storage of new study data in core. Because save area 0 contains the last displayed matrix image other than a save area matrix, the user can perform save area manipulations, such as reading other save areas into core and performing save area algebra, and then return to the original study in core by reading save area 0. Fifty-five optional save areas (10 through 64) can be specified by the user. However, these areas are restricted to matrices and have no provision for dynamic curves. When the user writes a matrix into a previously nonexistent optional save area, GAMMA-11 automatically produces the optional save area on the disk.

Save areas 1 through 9 each take up 53 blocks of the system disk. Each optional save area will reside on the system disk and take up 33 disk blocks each.

Save areas are RT-11 files. Their filenames are:

```
SVAR00.SYS     for save area 0
SVAR01.SYS     for save area 1
SVAR02.SYS     for save area 2
etc.
```

2.6.1 Save Area Descriptor Block

The first block (block 0) of the save area disk file is called the save area descriptor block. The save area descriptor block describes the type of save area (matrix or curve data) as well as the study with which the save area is currently associated. It contains the matrix type, number of frames, frame rate, pointers into the study, etc. Table 2-2 shows the layout of the save area descriptor block.

In Table 2-2, the variable types are abbreviated. ASC represents ASCII, INT represents integer, DPI represents double precision integer, and FP represents floating point.

Table 2-2
Save Area Descriptor Block

Decimal	Octal	Type	Name	Description
0	0	INT	NDXDEV	RAD50 device name of indexed device
2	2	INT	FILNAM	RAD50 file name and extension of file
8	10	INT	NPFILE	Number of patient files found on indexed device

Table 2-2 (Cont.)
Save Area Descriptor Block

Decimal	Octal	Type	Name	Description
10	12	INT	XTRBYT	Number of extra bytes in directory entry
12	14	INT	STYPE	Data type indicator (in low byte) 0 = no data in save area 1 = matrix data 200 = dynamic curves
14	16	INT	SDTYP	Save register number in low byte negative number=frame divide is set
16	20	ASC	SINDX	Index line (66 ASCII characters)
82	122	ASC	SCMDH	GAMMA-11 command string (46 ASCII characters)
128	200	INT	SXPND	Expand switch: 0 = no expansion, non-zero = expanded matrix
130	202	INT	SLADFG	Sliding add switch (the number of frames to add)
132	204	INT	SROTAT	Rotation factor: 0 = regular, 1, 2, or 3 to rotate axes
134	206	INT	SNESW	No enchancement switch
136	210	INT	SSD	Static or dynamic:0=static, non-zero=dynamic
138	212	INT	SDUAL	Dual isotope switch:0=no dual isotope 1 = isotope A, 2 = isotope B
140	214	INT	SORIG	Original study type (non-zero=list mode)
142	216	BYTE	SPOSOR	Position (rotation) switch
143	217	BYTE		Orientation switch
144	220	INT	SAQM	Acquisition mode: 1 = special, 2 = normal
146	222	INT	SFLDN	Flood correction switch: 0 = not done, 1 = flood cor. done
148	224	INT	SPDTA	Offset to data matrices
150	226	INT	SPPAD	Offset to previous admin block (multiple static only)
150	226	INT	SPZCT	Offset to Z-count block (dynamic only)
152	230	INT	SPNAD	Offset to next admin block (multiple static only)
152	230	INT	SPTOV	Offset to time of overflow block (dynamic only)
154	232	INT	SPADM	Offset to administrative data block
156	234	INT	SPCOM	Offset to comment block
158	236	INT	SDAD	Relative block number of present frame
160	240	INT	SDMOD	Isometric switch: 0 = intensity, 1 = isometrics
162	242	BYTE	STHSH	High threshold in %
163	243	BYTE		Step size in %
164	244	BYTE	STHSL	Low threshold in %

Table 2-2 (Cont.)
Save Area Descriptor Block

Decimal	Octal	Type	Name	Description
165	245	BYTE		Step size in %
166	246	INT	SSIZE	Number of words in current matrix
168	250	INT	SWDBYT	Word or byte switch: 0 = word, 1 = byte
170	252	INT	SDIM	Dimension size (32, 64, or 128)
172	254	INT	SMAX	Maximum cell count
174	256	INT	SMIN	Minimum cell count
176	260	DPI	SCOUNT	Total number of counts
180	264	INT	SMEAN	The average cell count
			STATIC MODE PARAMETERS	
206	316	INT	SMSCFR	Current frame number of multiple static study
208	320	DPI	SSTM	Duration of collection in seconds
212	324	DPI	SSVTM	Time of overflow in seconds
216	330	DPI	SSZCT	Z count, the number of events
220	334	INT	SSMSZ	Number of words in matrix
222	336	INT	SSFAD	Offset to first administrative data block
224	340	ASC	SVIEW	View of frame (10 ASCII characters)
			DYNAMIC MODE PARAMETERS	
206	316	INT	SCRFRM	Cummulative frame number
208	320	INT	SCURGP	Current group number
210	322	INT	SCURGF	Current number of frames in group
212	324	INT	SCURFM	Current frame within group
214	326	INT	SN	Total number of frames
234	352	INT	SG	Total number of groups
236	354	INT	SGROUP (1):SGPF	Number of frames in group
238	356	INT	SGPSZ	Number of words of frames in the group
240	360	INT	SGPCS	The close on overflow flag
242	362	INT	SGXTM (SGX)	Exposure rate: X frames per
244	364	INT	SGY	Y seconds (milliseconds for gated studies)
246	366	INT	SGROUP (2)	Group 2
			.	
			.	
			.	
356	544	INT	SGROUP (13)	Group 13
			ROI AND DYNAMIC CURVE PARAMETERS	
366	556	INT	NMROIS	The number of regions of interest (max of 12)

Table 2-2 (Cont.)
Save Area Descriptor Block

Decimal	Octal	Type	Name	Description
368	560	BYTE	ROIXY	Region of interest definition (1) X1 table If X1 and Y1 are negative, the region is undefined X1 = x-position of left ordinate
369	561	BYTE	Y1	Y1 = y-position of lower abscissa
370	562	BYTE	X2	X2 = right ordinate
371	563	BYTE	Y2	Y2 = upper abscissa
372	564	BYTE	ROIXY (2) X1	ROI table number 2
			.	
			.	
			.	
412	634	BYTE	ROIXY (12)	X1 ROI table 13
422	646	INT	IRM	Irregular ROI switch: 0 = regular region, non-zero = irregular
424	650	INT	SCELLS	The number of cells in the matrix
426	652	INT	NMCELLS (1)	Number of cells in region of interest 1
428	654	INT	NMCELLS (2)	Number of cells in ROI 2
			.	
			.	
			.	
448	700	INT	NMCELLS (12)	Cells in ROI number 12
452	704	FP	MAXCCR	Maximum cell count rate for the matrix
456	710	FP	MAXCCR (1)	Maximum cell count rate for region of interest 1
460	714	FP	MAXCCR (2)	Maximum cell count rate for ROI 2
			.	
			.	
			.	
500	764	FP	MAXCCR (13)	Maximum for ROI 12
			MATRIX SAVE AREA PARAMETERS	
452	704	DPI	ROICNT	Cell counts for matrix
456	710	DPI	ROICNT (1)	Cell counts for each ROI (used with matrix data)
460	714	DPI	ROICNT (2)	Cell counts for ROI 2
			.	
			.	
			.	
500	764	DPI	ROICNT (12)	Cell counts for ROI 12

2.6.2 Saving Matrix Data

Matrix data starts at block one of the save area disk file. If a specific save area contains matrix data, the data may use from 2 to 32 disk blocks, depending upon the size of the matrix.

Irregular region of interest (ROI) data is stored in the save area along with the matrix. Irregular ROIs are not applicable for 128x128 matrices. If the matrix size is 128x128, the matrix fills blocks 1 through 32. For 32x32 and 64x64 matrices, the matrix fills up to 16 blocks, and the irregular ROI information fills blocks 17 through 32. The irregular ROI information always starts at block 17, even if the matrix does not fill 16 blocks. Figure 2-5 shows the layout of a matrix save area.

The ROI information is stored an ROI map. The map contains one word per cell of the matrix; that is, each cell in the matrix is represented by one word in the ROI map. Figure 2-6 shows the layout of the ROI map compared to the cell map that shows on the display.

Each word in the ROI map defines, the ROI information for the corresponding cell in the display matrix. Each word in the ROI map contains one bit per ROI. Bits 0 through 11 represent ROIs A through L. If the corresponding cell is in ROI A, then bit 0 is set. If the cell is also within ROI B, then bit 1 is set, and so forth. Bits 12 through 15 are used internally and should not be written by a program. Figure 2-7 shows a word in The ROI map.

The ROI map is always a 64x64 matrix. Thus for a 32x32 matrix, four cells of the map are marked for each cell of the 32x32 matrix. Note on Figure 2-6, cells (1,1), (1,2), (2,1), and (2,2) would all be marked for the first cell of the 32x32 matrix.

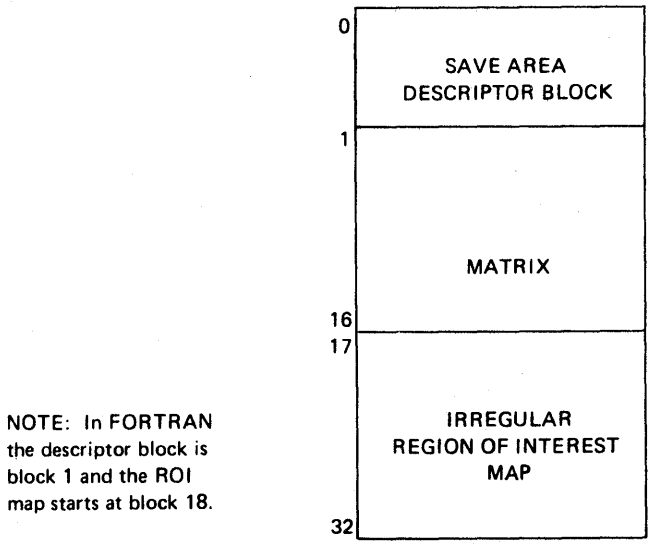


Figure 2-5 Matrix Save Area

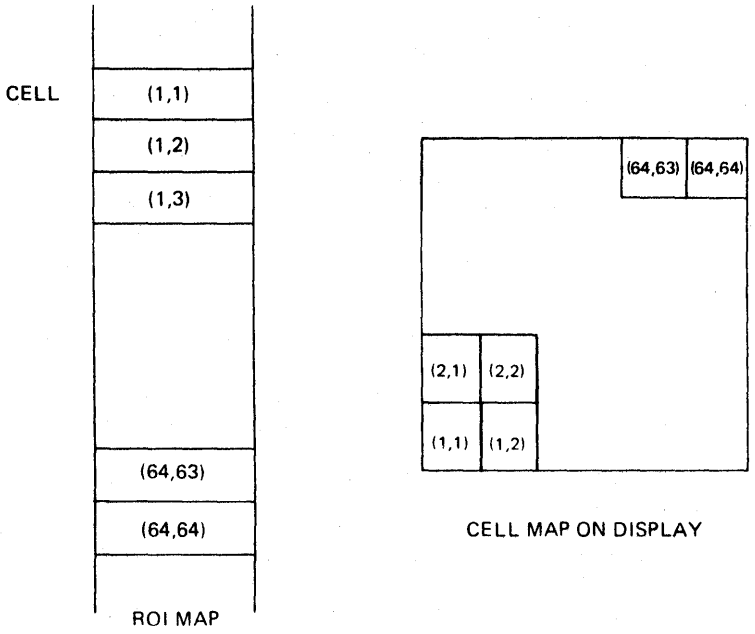


Figure 2-6 ROI Map and Cell Map

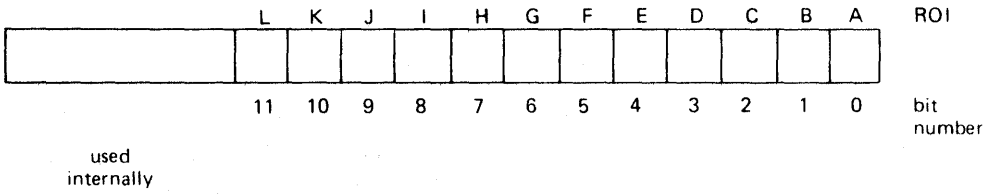


Figure 2-7 Word in ROI Map

2.6.3 Saving Dynamic Curves

Each dynamic curve uses four disk blocks and consists of up to 512 floating point numbers. A save area may hold up to thirteen dynamic curves, twelve representing the twelve possible regions of interest and the thirteenth representing the total count curve.

Blocks one through four of the dynamic curve data contain the total count curve which represents the total number of elements present within each frame of the study. Each following 4-block set contains the dynamic curve data for each region of interest. Therefore a save area containing twelve regions of interest uses all of the available 52 disk blocks. See Figure 2-8.

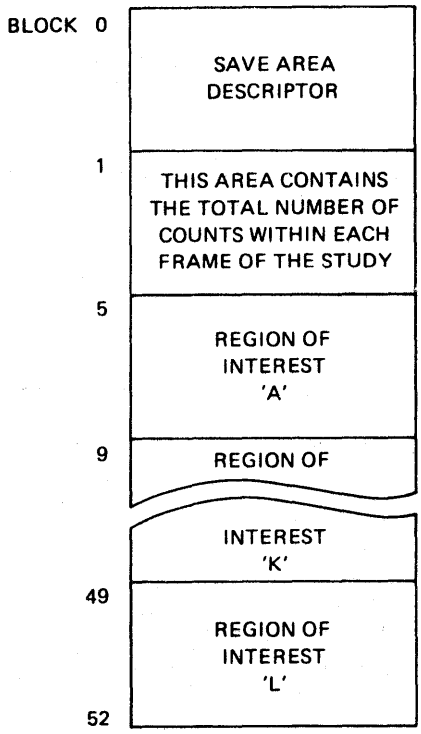


Figure 2-8 Dynamic Curve Save Area

2.7 INTERNAL GAMMA-11 FILES

GAMMA-11 requires a number of internal files for use as work areas and save areas. The naming conventions for these files are as follows:

SVARnn.SYS	is the name of a save area where nn is the number of the save area (00 through 64).
GAMMAx.SYS	names a work area where x is the identification character of the work area. This is the general form of the following internal files.
GAMMAD.SYS	used to store dynamic curves.
GAMMAS.SYS	a scratch file used to store intermediate values.
GAMMAI.SYS	a scratch file used by indexed display routine.
GAMMAP.SYS	the predefined study file.

GAMMAM.SYS	temporary storage for the irregular regions mark matrix (IR definition map). (Do not confuse this with the save area ROI map.)
GAMMAL.SYS	temporary storage for list mode parameters.
GAMMAC.SYS	storage for the color tables.
GAMMAB.SYS	dual display buffer

2.8 GAMMA-11 MACRO AND PLAYBACK FILES

GAMMA-11 macro filenames are all of the form

filename.GMC

where filename is the name given to the macro at its creation with the MC or MS command.

GAMMA-11 playback filenames are all of the form

filename.GPB

where filename is the name given to the playback file at its initialization with the PBI or PBM command.

2.8.1 Playback Files

A GAMMA-11 playback file (.GPB extension) consists of a one-block header block followed by the playback image buffers.

The first word of the playback header block contains the number of images stored in the playback file. To change the number of frames (in BASIC), declare the file an integer virtual array and change element 0 (zero).

The index line and comment line are stored as ASCII strings with maximum length of 128 characters. To change either the index line or comment line, declare the file a character virtual array of string length 128 and change element 1 (the index line) or element 2 (the comment line). You must make sure that the new string is less than 128 characters (0 to 127).

For example, the following BASIC program changes both the index line and the comment line. Note that this example deletes the comment line.

```
10 DIM #1, A$(3) = 128
20 OPEN 'filename.GPB' AS FILE #1
30 A$(1) = 'NEW INDEX LINE'
40 A$(2) = ''
50 CLOSE #1
60 END
```

Do not change any other elements in the file.

CHAPTER 3

BASIC AND FORTRAN SUPPORT

3.1 BASIC AND FORTRAN SUPPORT FOR GAMMA-11 F/B

The BASIC and FORTRAN support subroutines for GAMMA-11 allow complete access to the patient files and save areas (whether they hold matrix data or dynamic curves). The BASIC routines are linked with the BASIC interpreter and include a resident 1.5K buffer for data. The FORTRAN routines use the same resident 1.5K buffer for the data and perform the I/O similarly to BASIC. However, you have to link the object files of the FORTRAN routines with your compiled FORTRAN program to produce a running program.

Because the GAMMA-11 data, the BASIC interpreter or FORTRAN compiler, and a user-written program together need more memory than is available, the BASIC and FORTRAN support routines contain an automatic disk swapping routine. This swapping routine is transparent to the user, who can write programs as if there were more than enough memory.

There are two kinds of FORTRAN support routines discussed in this chapter. The first set of routines are called the FORTRAN support routines. These routines are similar to the BASIC routines.

The second set of FORTRAN routines are called the supplemental FORTRAN support routines. The supplemental routines do not perform input and output operations on patient files and save areas. The supplemental routines are not compatible with the FORTRAN support routines, and the two sets of FORTRAN routines can not be used together.

3.1.1 Support Routine Notation

The following table lists the notation for the parameters of the BASIC and FORTRAN support subroutines.

NOTE

Since BASIC numeric variables have no type (e.g., integer or real number), the last column of this table is applicable to FORTRAN only.

Variable Name	Description	FORTRAN Variable Type
isanum	represents a save area	integer
isatype	represents a save area type (matrix or dynamic curves)	integer
index	represents an index number which refers to a specific element of a save area or patient file. For example, the index number of the patient name is 1 and of the patient number is 2.	integer
i,j	represent the row (i) and column (j) indices of a matrix. Note that row 1, column 1 is the lower left corner of the matrix.	integer,integer
iframe	represents a frame number	integer
icurve	represents a dynamic curve number	integer
ipoint	represents a point number from a dynamic curve	integer
value	represents the value in an element of the data file or save area	all value types within administrative and save area descriptor blocks are given in Tables 2-1 and 2-2. Points on a dynamic curve are floating point.
string	represents the equivalent of value if the element is an ASCII string	this variable should be a logical array in FORTRAN.
dev:file.Xnn	represents the RT-11 file descriptor of a patient file	this descriptor should be contained in a logical array in FORTRAN
[,len]	represents the optional length of the logical array given by "string" above. This parameter is valid in FORTRAN only.	integer

3.1.2 Patient Data File Subroutines

The following subroutines reference the patient data files. Table 3-1 shows the administrative data block layout with the indexes needed for subroutines GPAR, GPAW, GPDR, and GPDW.

Subroutine	Explanation
GPFR('dev:file.Xnn') or GPFR(string) or GPFR(string[,len])	Opens a patient file for read only processing. When a file is opened with this subroutine call, it cannot be modified. BASIC examples: CALL GPFR('RK1:NAME.X00') CALL GPFR(V$) However, if an ASCII string V$ is used, then V$='RK1:NAME.X00' must be defined before GPFR(V$) is called. FORTRAN examples CALL GPFR('RK1:JOHNDO.X01') CALL GPFR(VA) CALL GPFR(VA,6) In these examples, VA is a logical array, and 6 is the length of the logical array. The 6 is optional.
GPFW('dev:file.Xnn')	Opens a patient file for read or write processing. When a file is opened with this subroutine call, it can be modified. The alternate forms of the call and examples are similar to GPFR.
GPF()	Closes a patient file that is currently open. This subroutine should be used to ensure that all modifications to a file have been made.
GPAR(index,string[,len])	Returns in parameter string the ASCII string in element(index) from the administrative data block. The parameter [,len] is the optional array length for the logical array in FORTRAN.
GPAW(index,string[,len])	Stores the ASCII string in element(index) of the administrative data block of the patient file.
GPDR(index,value)	Returns in value the value of element(index) in the administrative data block.
GPDW(index,value)	Stores value in the administrative data block as element(index).
GPMR(iframe,i,j,value)	Returns in value the value of element(i,j) of frame iframe of a study. This subroutine may only be used for static and dynamic studies.
GPMW(iframe,i,j,value)	Stores value as the element(i,j) of frame iframe. This subroutine may be used only for static and dynamic studies.
GPLR(n,x,y,t,g)	Returns from list mode element(n), the following values in the variables: x = X-coordinate y = Y-coordinate t = 0, if there is no time mark = 1, if the time mark is set g = 0, if there is no gate mark = 1, if the gate mark is set

Subroutine	Explanation
	Note that for GPLR and GPLW, x, y, t, and g are integers (FORTRAN only).
GPLW(n,x,y,t,g)	Stores whatever is in x, y, t, and g into the list mode element number (n).

Table 3-1
Administrative Data Block

Ascii String Variable Table (FORTRAN and BASIC)
--- Subroutines GPAR and GPAW ---

Index	Name	Description
1	PATNAM	Patient name
2	PATNUM	Patient number
3	ATIME	Acquisition time (supplied by program)
4	ADATE	Acquisition data (supplied by program)
5	BIRTHD	Patient birth date
6	DOC	Doctors name
7	ORGAN	Organ being studied
8	VIEW	View of picture
9	CMTRT	Collimator type
10	AQMODE	Acquisition mode: 1=special, 2=normal
11	ISOTOP	Isotope being used
12	DOSE	Dosage
13	ISO2	2nd isotope being used
14	DOSE2	2nd dosage (dual isotope study)
15	ISMODE	1=single isotope, 2=dual isotope
16	ORIENT	Orientation switch
17	POSSWT	Position (rotation) switch
18	AMACRO	Auto analysis macro name
19	AUTO	Auto analysis switch (y or n)
	GATE-SYNCHRONIZED ACQUISITION (GSA) COLLECTION PARAMETERS	
20	CAMID	Camera number (0-3) (NCV11 only)
21	GSAMTX	GSA matrix type (1 or 2)
22	GSAFRM	Number of frames
23	GSADUR	Frame duration in msec
24	GSATOL	Tolerance in msec
25	GSAEFM	End Frame (1, 2, or 3) 1 = Time 2 = Counts 3 = Cycles
31	GSAPSC	Preset counts
32	GSACYC	Preset cycles
33	GSAMIN	Preset minutes
34	GSASEC	Preset seconds
	STATIC AND LIST COLLECTION PARAMETERS	
26	ENDFRA	Method of ending study: 1=time, 2=counts
27	MINUTE	The number of minutes in the study
28	SECOND	The number of seconds in the study
29	PSCNT	The number of preset counts chosen

Table 3-1 (Cont.)
Administrative Data Block

Index	Name	Description
		STATIC MODE COLLECTION PARAMETERS
36	SMTXSZ	The type of matrix (1, 2, 3, 4, or 5)
37	SMTXCS	Close on overflow: <space>=do not close, <t>=close
38	MSFRM	The number of frames
		LIST MODE COLLECTION PARAMETERS
41	LDBLCK	The number of disk blocks of data
42	LDBPC	Method of closing: 1=by counts, 2=by number of blocks
43	LDELST	Method of starting: y=delayed start, n=immediate start
44	LDELRT	Count rate for delayed start (maximum of 20,000)
45	LGSA	If 'y', study in a gated list mode
		DYNAMIC MODE COLLECTION PARAMETERS
51	GRP1	The number of frames in group 1
52		The type of matrix(1, 2, 3 or 4)
53		The type of close
54		Frame rate: x frames
55		Per y seconds (milliseconds for gated)
56	GRP2	Group 2 (same 5 paramenters as group 1)
61	GRP3	Group 3
	.	
	.	
131	GRP13	Group 13

Data Value Table (FORTRAN and BASIC)
--- Subroutines GPDR and GPDW ---

1	COLTYP	Collection type Second byte < 0 list mode = 0 dynamic study > 0 static study
2	COMPRS	Offset to comment block
3	TOTBLK	Total number of blocks in study
4	FADOFF	Offset to first admin block (multiple static)
5	DATTYP	Data type: 0=patient data, 1=flood
6	MDOFF	Offset to matrix data
7	PADOFF	Offset to previous admin block (multiple static)
7	ZCTOFF	Dynamic: offset to z count block
8	NADOFF	Offset to next admin block (multiple static)
9[1]	ZCOUNT	Z count, the number of events (double precision integer)
10[1]	OVFTIM	Time of overflow clock counter
11	CFRM	Current frame number
12	FRAMEN	Total number of frames
13	GROUPN	Total number of groups

Table 3-1 (Cont.)
Administrative Data Block

Index	Name	Description
14	BINSEC	Collection time in seconds
15	BINCYC	# cycles collected
16	BINZLO BINZHI	Total counts collected
17	BINDUR	Frame duration in msec
18	BINTOL	Tolerance in msec
19	BINBAD	# rejected cycles

[1] Format of data is double precision integer (FORTRAN data type REAL*4)

3.1.3 Save Area Subroutines

Only one save area can be opened for reading at a time with GSAR. However, with GSAW, you can write to any of the save areas--opened or unopened. The term current save area refers to the save area that is currently open.

Table 3-2 shows the layout of the save area descriptor block and the indexes into the block needed by the save area subroutines.

The following subroutines access the Save Areas.

Subroutine	Explanation
GSAR(isanum,isatype)	Opens save area isanum; The subroutine returns the save area type in isatype. isatype=1 if the save area contains matrix data; isatype = -1 if the save area contains dynamic curves; isatype = 0 if the save area contains neither matrix data nor dynamic curves or if the save area does not exist.
GSVG(index,value)	Returns the value of element(index) of the save area descriptor block.
GSVP(index,value)	Stores value in element(index) of the save area descriptor block.
GMXG(i,j,value)	Returns the value of element(i,j) of the matrix.
GMXP(i,j,value)	Stores value as element(i,j) of the matrix.
GCVG(icurve,ipoint,value)	Returns the value of point ipoint of curve icurve.
GCVP(icurve,ipoint,value)	Stores value as point ipoint of curve icurve.
GDIS(icurve)	Plots curve icurve on the display.

Subroutine	Explanation
GPOV(icurve)	Plots curve icurve on the display, overlaying the previously displayed curve.
GPKX(x) [BASIC ONLY]	Displays a cursor above a point plotted by GDIS or GPOV and waits for the user to pick a point. If the user types an "L" or an "R", the cursor moves to the left or the right, respectively. If the user types a "J", the cursor moves 10 spaces in the direction last typed. When the user types an "M", the x value (point number) returns as x.
GPKY(y) [BASIC ONLY]	Same as subroutine GPKX except the y value (count rate) is returned.
FGPICK (ix,y) [FORTRAN ONLY]	Displays a cursor on the display above a point plotted by GDIS or GPOV. If the user types an "L" or an "R", the cursor moves to the left or the right respectively. If the user types a "J", the cursor moves 10 spaces in the direction last typed. When the user types an "M", FGPICK returns the position of the cursor in ix (integer) and y (real).
GSAW(isanum)	Writes the current save area into save area number isanum.
GSAG(index,string)	Returns in string ASCII element(index) of the save area descriptor block.
GASP(index,string)	Stores string as the ASCII element(index) of the save area descriptor block.

Table 3-2
Save Area Descriptor Block

Save Area String Table (FORTRAN and BASIC)
--- Subroutines GSAG and GASP ---

Index	Name	Description
1	SINDX	Index line (66 ASCII characters)
2	SCMDH	GAMMA-11 command string (46 ASCII characters)
3	SVIEW	View of frame (10 ASCII characters)

Save Area Data Table (FORTRAN and BASIC)
--- Subroutines GSVG and GSVP (FORTRAN and BASIC) ---

1	NDXDEV	RAD50 device name of indexed device 2-4 FILNAM RAD50 file name and extension of file (3 integers)

Table 3-2 (Cont.)
Save Area Descriptor Block

Index	Name	Description
5	NPFILE	Numbers of patient files found on indexed device
6	XTRBYT	Number of extra bytes in directory entry
7	STYPE	Data type indicator (in low byte) 0 = no data in save area 1 = matrix data 200 = dynamic curves
8	SDTYP	Save register number in low byte negative number=frame divide is set
9	SXPND	Expand switch: 0=no expansion, non-zero= expanded matrix
10	SLADFG	Sliding add switch (the number of frames to add)
11	SROTAT	Rotation factor: 0=regular, 1, 2, or 3 to rotate axes
12	SNESW	No enhancement switch
13	SSD	Static or dynamic: 0=static, non-zero=dynamic
14	SDUAL	Dual isotope switch: 0=no dual isotope 1 = isotope A, 2 = isotope B
15	SORIG	Original study type (non-zero=list mode)
16[1]	SPOSOR	Position (rotation) switch
17[1]		Orientation switch
18	SAQM	Acquisition mode: 1=special, 2=normal
19	SFLDN	Flood correction switch: 0=not done, 1=flood correction done
20	SPDTA	Offset to data matrixes
21	SPPAD	Offset to previous admin block (static only)
21	SPZCT	Offset to z-count block (dynamic only)
22	SPNAD	Offset to next admin block (static only)
22	SPTOV	Offset to time of overflow block (dynamic only)
23	SPADM	Offset to administrative data block
24	SPCOM	Offset to comment block
25	SDAD	Relative block number of present frame
26	SDMOD	Isometric switch: 0=intensity, 1=isometrics
27[1]	STHSH	High threshold in %
28[1]		Step size in %
29[1]	STHSL	Low threshold in %
30[1]		Step size in %
31	SSIZE	Number of words in current matrix
32	SWDBYT	Word or byte switch: 0=word, 1=byte
33	SDIM	Dimension size (32, 64, or 128)
34	SMAX	Maximum cell count
35	SMIN	Minimum cell count
36[2]	SCOUNT	Total number of counts
37	SMEAN	The average cell count

Table 3-2 (Cont.)
Save Area Descriptor Block

STATIC MODE PARAMETERS

Index	Name	Description
121	SMSCFR	Current frame number of multiple static study
122[2]	SSTM	Duration of collection in seconds
123[2]	SSVTM	Time of overflow in seconds
124[2]	SSZCT	Z count, the number of events
125	SSMSZ	Number of words in matrix
126	SSFAD	Offset to first administration block

DYNAMIC MODE PARAMETERS

41	SCRFRM	Cumulative frame number
42	SCURGP	Current group number
43	SCURGF	Current number of frames in group
44	SCURFM	Current frame within group
45	SN	Total number of frames
50	SG	Total number of groups
51	SGROUP(1):SGPF	Number of frames in group
52	SGPSZ	Number of words of frames in the group
53	SGPCS	The close on overflow flag
54	SGXTM (SGX)	Exposure rate: x frames per
55	SGY	Y seconds (milliseconds for gated)
56	SGROUP(2)	Group 2
	.	
	.	
111	SGROUP(13)	Group 13

ROI AND DYNAMIC CURVE PARAMETERS

130	NMROIS	The number of regions of interest (max of 12)
131	ROIXY(1) X1	Region of interest definition table If X1 and Y1 are negative, the region is undefined X1 = x-position of left ordinate
132[1]	Y1	Y1 = y-position of lower abscissa
133[1]	X2	X2 = right ordinate
134[1]	Y2	Y2 = upper abscissa
135[1]	ROIXY(2) X1	ROI table number 2
	.	
	.	
175[1]	ROIXY(12) X1	ROI table 12
183	IRM	Irregular ROI switch: 0= regular region, non-zero=irregular
186	SCELLS	The number of cells in the matrix
187	NMCELLS(1)	Number of cells in region of interest 1
188	NMCELLS(2)	Numbers of cells in ROI 2
	.	
	.	

Table 3-2 (Cont.)
Save Area Descriptor Block

Index	Name	Description
198	NMCELLS(12)	Cells in ROI number 12
201[3]	MAXCCR	Maximum cell count rate for the matrix
202[3]	MAXCCR(1)	Max.cell count rate for ROI 1
203[3]	MAXCCR(2)	Max.cell count rate for ROI 2
	.	
	.	
213[3]	MAXCCR(12)	Max.for ROI 12
	MATRIX SAVE AREA PARAMETERS	
201[2]	ROICNT	Cell counts for matrix
201[2]	ROICNT(1)	Cell counts for each ROI (used with matrix data)
203[2]	ROICNT(2)	Cell counts for ROI 2
	.	
	.	
213[2]	ROICNT(12)	Cell counts for ROI 12

[1] Byte data (FORTRAN uses INTEGER*2 format)
[2] Double precision integer data (FORTRAN uses REAL*4 format)
[3] Floating point data (FORTRAN uses REAL*4 format)

3.1.4 General Purpose Support Subroutines for BASIC and FORTRAN

GAM(string)

Exits from BASIC or FORTRAN and loads the background GAMMA-11 program. BGAMMA is executed, and it interprets the string as the first command. If string is null, the background command table is displayed. If string is an illegal command, an error message is displayed and typing a carriage return will return GAMMA-11 to the command table.

GCHR(string[,lineno,icolno])

Prints the character string (string) starting at location (lineno,icolno) on the VSV01. The parameters lineno and icolno are optional. If lineno = negative number or zero (0), the subroutine erases the screen. If you call GCHAR with only the string parameter (e.g. GCHAR(string)), the string is printed starting at the current cursor position. (NOTE: This subroutine works for the VSV01 display only.) You can reference only line numbers 1 through 25, and columns 1 through 64.

3.1.5 Linking FORTRAN Subroutines with a User Program

For you to use the FORTRAN/GAMMA-11 subroutines with your own program, you must link four FORTRAN object modules to your program. The four object modules are:

```
GMFOR1.OBJ
GMFOR2.OBJ
GMFOR3.OBJ
GMFERR.OBJ
```

GMFOR1.OBJ and GMFOR2.OBJ should always be linked with a user program whenever any of the support routines are referenced in the user program. You should link GMFOR3.OBJ when any curve plotting subroutines are used. You should always link GMFERR.OBJ because it contains the FORTRAN error messages.

NOTE

You can install the FORTRAN OTS library in SYSLIB.OBJ or in FORLIB.ORJ. See Section 2.4.1 of the RT-11 FORTRAN IV Installation Guide (DEC-11-LRSIA-A-D). If you have installed the FORTRAN OTS library in SYSLIB.OBJ, you do not need the FORLIB or /F parameters when you link your FORTRAN program.

If a program named PGM references only GAMMA-11 patient files, type:

```
.LINK PGM,GMFOR1,GMFOR2,GMFERR,FORLIB
```

If a program references save area data and plot curves, type:

```
.LINK PGM,GMFOR1,GMFOR2,GMFOR3,GMFERR,FORLIB
```

If the overlay feature of the RT-11 Linker is used, GMFOR1 should be linked to the root section of the program. GMFOR2, GMFOR3, and GMFERR can be included in the overlays if desired. To use the overlay feature, type:

```
.R LINK
*PGM=PGM,GMFOR1,FORLIB/C
*GMFOR2/0:1/C
*GMFOR3/0:1/C
*GMFERR/0:1
```

3.1.6 BASIC And FORTRAN Error Messages

The error messages are the same for both the BASIC and FORTRAN support subroutines. However, the format of the messages vary between BASIC and FORTRAN.

The format for the BASIC error messages is:

```
?GAMMA-F-Save are numbers too large or negative AT LINE 20
```

The line number of the line where the error occurred is given. In this example, the error occurred at line 20. In BASIC, the system returns to a READY when an error occurs.

The format for the FORTRAN error messages is:

```
?GAMMA-F-Save area number too large or negative
?Err 0 Non-FORTRAN error call
in routine ".MAIN." line 5
```

The first line of the message states the problem. The second and third lines of the message state the routine name and line in which the error occurred. In this example, the error occurred in line 5 of the main program. In FORTRAN, the system returns to the RT-11 monitor when an error occurs.

The error messages for the BASIC and FORTRAN support subroutines are listed below.

Null file name

Routines GPFR, GPFW
A null string was given as the GAMMA patient file name.

Illegal device

Routines GPFR, GPFW
An illegal device name was given in the string while opening the GAMMA patient file.

Illegal file name

Routines GPFR, GPFW
Illegal RAD50 character was given as part of the GAMMA-11 patient file name.

Non-file structured device

Routines GPFR, GPFW
A non-file structured device (e.g., paper tape, line printer) was given in the string while opening the GAMMA patient file.

No device handler loaded

Routines GPFR, GPFW
The device handler is not in memory, and the GAMMA patient file cannot be opened.

GAMMA file lookup error

Routines GPFR, GPFW, GSAR
A lookup error occurs trying to open a patient file or a save area file. This error usually means that the file is not on the device specified.

GAMMA file not open

All routines except GPFR, GPFW, GSAR
Subroutine tried to reference a GAMMA patient file or save area file before it was opened.

GAMMA file savestatus error

Routines GPFR, GPFW, GPF, GPMR, GPMW, GPLR, GPLW, GSAR, GMXG, GMXP, GCVG, GCVP, GSAW
Save status error occured during input/output operation (probable hardware error).

GAMMA file reopen error

Routines GPFR, GPFW, GPF, GPMR, GPMW, GPLR, GPLW, GSAR, GMXG, GMXP, GCVG, GCVP, GSAW
Reopen error during an input/output operation (probable hardware error).

GAMMA file read error

Routines GPFR, GPFW, GPF, GPMR, GPMW, GPLR, GPLW, GSAR GMXG, GMXP, GCVG, GCVP, GSAW
Read error during I/O (probable hardware error).

GAMMA file write error

Routines GPFR, GPFW, GPF, GPMR, GPMW, GPLR, GPLW, GSAR, GMXG, GMXP, GCVG, GCVP, GSAW
Write error during I/O (probable hardware error).

Frame number too large or negative

Routines GPMR, GPMW
The frame number (iframe) is not within the range of the number of frames in the current patient file.

List element too large or negative

Routines GPLR, GPLW
The list mode element (n) is outside the boundaries of the number of elements in the patient file.

Index too large or negative

Routines GPAR, GPAW, GPDR, GPDW, GSVG, GSVP, GSAG, GSAP
The index number (index) exceeds the number of indices in the table that is being referenced.

Curve number too large or negative

Routines GCVG, GCVP, GDIS, GPOV
The curve number (icurve) is greater than 12, the maximum number of curves in a save area.

Point number too large or negative

Routines GCVG, GCVP
The point number (ipoint) is outside the boundaries of the number of points in the save area curves.

Dimension too large or negative

Routines GPMR, GPMW, GMXG,
The element specified by (i,j) of the patient file or save area is outside the boundaries of the matrix.

Save area number too large or negative

Routines GSAR, GSAW
The save area number (isanum) referenced exceeds 64, the maximum number of save areas.

Curve save area number too large or negative

Routines GSAW
An attempt was made to write a dynamic curve Save Area in a save area number (isanum) greater than 9.

Illegal parameter value

Routines All routines except GPFR, GPFW, GPF
An illegal value (less than or equal to zero) is set for a subroutine parameter (e.g., index number (index), matrix dimension (i,j), point number (ipoint), frame number (ifame)).

3.1.7 BASIC and FORTRAN Examples

BASIC Example 1

The following example reads a patient summary and then prints it out. The important lines of this example are lines 20 and 30. In line 20, the user enters the patient file name which is stored in variable A$. In line 30, that file is opened for reading and writing.

```
10 REM -- READ IN PATIENT FILE --
20 PRINT 'INPUT PATIENT FILE NAME' \ INPUT A$
30 CALL GPFW(A$)
40 PRINT
50 REM -- OUTPUT FILE SUMMARY --
60 FOR I=1 TO 20
70 PRINT '*'; \ NEXT I
80 PRINT  \ PRINT
90 PRINT 'PATIENT:'; \ FOR I=1 TO 8
100 CALL GPAR(I,B$) \ PRINT B$
110 NEXT I
120 CALL GPDR(12,B)
130 PRINT 'NUMBER OF FRAMES ='; B
140 PRINT
150 FOR I=1 TO 20
160 PRINT '*'; \ NEXT I
170 CALL GPF()
180 END

READY
```

BASIC Example 2

The following example integrates a dynamic curve. The user picks the save area where the curve is stored, the curve to be displayed, and the left and right bounds of the integral. Since the raw counts are stored with the curve data, the integral is merely a summation of the counts between the boundaries chosen (using subroutine GCVG in line 110).

```
20 REM -- READ DYNAMIC CURVE SAVE AREA --
30 PRINT 'WHICH SAVE AREA'; \ INPUT A1
35 CALL GSAR(A1,A)
40 REM -- READ CURVE INTO THE BUFFER --
50 PRINT 'WHICH CURVE TO BE DISPLAYED'; \ INPUT B1
55 CALL GDIS(B1)
60 REM -- FIND THE NUMBER OF POINTS --
70 CALL GSVG(41,B)
80 REM -- FIND THE INTEGRAL OF THE CURVE
90 I=0
100 PRINT 'THERE ARE'; B; 'POINTS'
102 PRINT 'PICK THE BOUNDS OF THE INTEGRAL'
104 CALL GPKX(D) \ CALL GPKX(E)
106 FOR J=D TO E
110 CALL GCVG(B1,J,K)
120 I=I+K
130 NEXT J
140 PRINT 'INTEGRAL OF CURVE'; B1; '='; I
150 END

READY
```

BASIC Example 3

The following example initializes a save area to hold static matrix data. The user picks the save area to be initialized. Then that save area is opened as a virtual file (lines 40, 50, and 60) to allow the program to zero the save area descriptor block (lines 80 and 90). The user chooses the type of matrix and the subroutine initializes the save area descriptor block. Finally, in lines 160 to 200, the matrix is filled (with whatever you choose). In this example, an "X" is put in the matrix. Then the BASIC program returns to the GAMMA-11 data analysis program to continue analysis.

```
10 REM -- ZERO SAVE AREA DESCRIPTOR BLOCK --
20 REM
30 PRINT  \ PRINT 'SAVE AREA MATRIX INITIALIZATION'
40 PRINT  \ PRINT 'WHICH SAVE AREA'; \ INPUT A3$ \ LET A3=VAL(A3$)
50 LET A$='SVAR0'&A3$&'.SYS'
60 DIM #1,F1(255)
70 OPEN A$ AS FILE 1
80 FOR I=0 TO 255 \ LET F1(I)=0 \ NEXT I
90 CLOSE 1
100 REM
110 REM -- CHOOSE MATRIX TYPE, THEN INITIALIZE
120 REM
130 PRINT 'WHAT MATRIX SIZE:(32,64,128)'; \ INPUT A1
140 PRINT 'BYTE OR WORD:(1=BYTE,0=WORD)/'; \ INPUT A2
150 CALL GSAR(A3,Z) \ GOSUB 210
160 FOR I=1 TO A1 \ LET J=A1+1-I
170 CALL GMXP(I,I,I) \ CALL GMXP(I,J,J)
180 NEXT I
190 CALL GSAW(A3)
200 CALL GAM('CA')
210 REM
220 REM -- MATRIX INITIALIZATION SUBROUTINE
230 REM SET TO INIT A STATIC MATRIX
240 REM -- A1=SIZE(32,64,128)..A2=BYTE(1)..OR WORD (0)
250 REM
260 CALL GSVP(7,1) \ CALL GSVP(8,A3)
270 CALL GSVP(27,100) \ CALL GSVP(28,5)
280 CALL GSVP(29,0) \ CALL GSVP(30,5)
290 LET A4=512
300 IF A1=32 THEN 320 \ LET A4=A4*4
310 IF A1=64 THEN 320 \ LET A4=A4*4
320 IF A2=1 THEN 330 \ LET A4=A4*2
330 CALL GSVP(31,A4) \ CALL GSVP(125,A4)
340 CALL GSVP(32,A2) \ CALL GSVP(33,A1)
350 CALL GASP(1,'GAMMA-11 SAVE AREA')
360 RETURN
370 END

READY
```

BASIC Example 4

The following macro creates the playback file GSA.GPB, plays the playback, calls the BASIC program NEWNME (line 4), and then plays the playback file a second time. Program NEWNME changes the patient file index line in the playback file called GSA.GPB. When the program is finished, the macro will continue execution and replay the playback.

MACRO

```
1)  PBI GSA,0,48,1
2)  RS0;BE;LT10
3)  48:PBS;!SK
4)  PB GSA;BA NEWNME
5)  PB GSA
```

BASIC PROGRAM NEWNME

```
10 DIM #1,A$(16)=64
20 OPEN 'GSA.GPB' AS FILE #1
30 PRINT 'OLD PATIENT INDEX:';A$(2)
40 PRINT 'INPUT NEW INDEX:'; \ INPUT B$
50 LET A$(2)=B$
60 CLOSE #1
70 CAL GAM('CA')
```

FORTRAN Example 1

The following FORTRAN example is similar to BASIC Example 3. This program initializes a save area to hold static matrix data. The main program asks the user for the save area number, opens the save area, initializes the save area descriptor block (subroutine INITMA), and fills the matrix (subroutine FILLMA). Subroutine IALPH converts the numeric save area number to ASCII data for the save area name.

```
      INTEGER NAME(6),ADMIN(256)
1     FORMAT(I2)
      NAME(1)='SV'
      NAME(2)='AR'
      NAME(4)='.S'
      NAME(5)='YS'
      NAME(6)=0
      TYPE *, 'SAVE AREA MATRIX INITIALIZATION'
      TYPE *, 'WHICH SAVE AREA?'
      ACCEPT *, NUM
      ENCODE(2,1,NAME(3)) NUM
      OPEN(UNIT=1,NAME=NAME,ACCESS='DIRECT',TYPE='UNKNOWN',
     1RECORDSIZE=128,INITIALSIZE=33,ASSOCIATEVARIABLE=N1)
      DO 100 I=1,256
100   ADMIN(I)=0
      WRITE(1'1)ADMIN
      WRITE(1'33)ADMIN
      CLOSE(UNIT=1)
      TYPE *, 'WHAT MATRIX SIZE:(32,64,128)'
      ACCEPT *, NSIZE
      TYPE *, 'BYTE OR WORD:(1=BYTE,0=WORD)'
```

```
      ACCEPT *, NTYPE
      CALL GSAR(NUM,M)
      CALL INITSA(NSIZE,NTYPE,NUM)
      CALL INITMA(NSIZE)
      CALL GSAW(NUM)
      CALL BGAMMA('CA')
      STOP
      END
C
      SUBROUTINE INITSA(NSIZE,NTYPE,NUM)
      CALL GSVP(7,1)
      CALL GSVP(8,NUM)
      CALL GSVP(27,100)
      CALL GSVP(28,5)
      CALL GSVP(29,0)
      CALL GSVP(30,5)
      I=512
      IF(NSIZE.EQ.64)I=I*4
      IF(NSIZE.EQ.128)I=I*16
      IF(NTYPE.EQ.0)I=I*2
      CALL GSVP(31,I)
      CALL GSVP(125,I)
      CALL GSVP(32,NTYPE)
      CALL GSVP(33,NSIZE)
      CALL GASP(1,'GAMMA-11 SAVE AREA',22)
      RETURN
      END
C
      SUBROUTINE INITMA(NSIZE)
      DO 100 I=1,NSIZE
      J=NSIZE+1-I
      CALL GMXP(I,I,I)
100   CALL GMXP(I,J,J)
      RETURN
      END
```

3.2 SUPPLEMENTAL FORTRAN SUPPORT

Besides the FORTRAN support routines listed in Section 3.1, other FORTRAN routines exist to access patient files and save areas, and to plot dynamic curves. The routines listed in this section do not perform input and output operations on the patient files and save areas. To use these supplemental routines, you must first assign and define the input and output files as random access files and then read the appropriate blocks into arrays before calling the routines. Refer to Tables 3-1 and 3-2 for a description of the internal structure of the patient files and save areas.

3.2.1 FORTRAN and GAMMA-11 Variables

The variables used in the GAMMA-11 files are not compatible with FORTRAN IV. The GAMMA-11 variables are unsigned (i.e., not 2's complement) numbers. FORTRAN IV variables must be signed.

Below is the notation that is used to denote the GAMMA-11 and FORTRAN variable types.

GAMMA-11 Variables

Name	Data Type	Contents
g8	LOGICAL*1	Unsigned, 8-bit datum
g16	INTEGER*2	Unsigned, 16-bit datum
g32	REAL*4 or INTEGER*4	Unsigned, 32-bit double precision integer.

FORTRAN Variables

integerf	INTEGER*2	Signed integer
realf	REAL*4	Real, floating point number

3.2.2 Arrays

When you use this set of FORTRAN support routines, you must handle inputting and outputting the files yourself. The following array notation describes the format for handling these files.

Name	Description
rawfile	A 512-byte array in which you load the entire administrative data block of a patient study (using a direct-access read).
patientinfo	A real array, dimensioned (3,42), which contains ASCII data converted from rawfile. The patientinfo array is obtained from the rawfile array by using subroutine FGADM1.
ipointers	A 75-word integer array which contains pointers and parameters from the administrative data block contained in array rawfile. The pointers array is obtained from array rawfile by using subroutine FGADM1.

Name	Description
rawcomments	A 510-byte array which contains the comment block of the patient study (read by a direct-access read).
comments	A 510-byte logical array dimensioned (51,10) which contains the ASCII text of the comment block. The array comments is obtained from the array raw comments using subroutine FGCOM1. Each position of the array is one ASCII character.
savearea	A 256-word array containing the descriptor block of a save area (read by a direct-access read).
curve	A 512-real element array containing a dynamic curve read from a save area (read by a direct-access read).

3.2.3 Functions

The following functions convert unsigned integer data from GAMMA-11 to signed integer or floating point format of FORTRAN IV.

IBYTE(g8)	Returns the byte datum, g8, as a signed integer, integerf.
RSPI(g16)	Returns the 16-bit unsigned integer, g16, as a floating point number, realf.
RDPI(g32)	Returns the unsigned 32-bit integer, g32 as floating point number, realf.

The following functions convert FORTRAN IV data to GAMMA-11 format.

LBYTE(integerf)	Returns the signed integer, integerf, as an unsigned 8-bit integer, g8.
ISPR(realf)	Returns the floating point number, realf, as an unsigned 16-bit integer, g16.
RDPR(realf)	Returns the floating point number, realf, as an unsigned 32-bit integer, g32.

GAMMA-11 word data does not have to be converted to real format unless the number of cell counts exceeds 32767. In a typical study, such large cell counts are extremely unlikely.

3.2.4 Subroutines

The following subroutines process GAMMA-11 patient studies.

FGADM1(rawfile,pointers,patientinfo)
Converts the patient administrative data block in array rawfile into the ASCII array patientinfo and parameter pointer block pointers. If patientinfo is not specified, no ASCII data is converted. Rawfile and patientinfo, or rawfile and pointers cannot be equivalenced.

FGADM2(rawfile,pointers,patientinfo)
The ASCII array patientinfo and parameter pointer block pointers are converted into a patient administrative data block in array rawfile. Rawfile and patientinfo, or rawfile and pointers cannot be equivalenced.

FGCOM1(rawcomments,comments)
Converts the comment block (in rawcomments) into a FORTRAN ASCII array comments.

FGCOM2(rawcomments,comments)
Converts the FORTRAN ASCII array comments into a GAMMA-11 comment block rawcomments. Rawcomments and comments can be equivalenced.

IFGFRM(ipointers,i)
A function which returns the record number of frame i. The array ipointers is the pointer array set up by FGADM1.

IGLSTR(n,x,y) Return the x and y coordinates of the list mode element n. The function returns 1 of 4 possible values.

```
 2  Time mark not set, gatemark set.
 1  No time mark, no gatemark.
-1  Time mark set, gatemark not set.
-2  Time mark set, gatemark set.
```

Note that for IGLSTR and IGLSTW, x, y, t, and g are integers.

IGLSTW(x,y,t,g)
A function which returns a list mode element number from the four parameters, x, y, t, and g.

NOTE

FORTRAN record numbers are one greater than RT-11 block numbers.

The following subroutines reference save area data.

FGPLOT(savearea,curve) Plots a dynamic curve on the display. The array savearea contains the save area descriptor block and curve is an array that contains the floating point dynamic curve data.

FGPICK(ix,y) Displays a cursor on the display above a point already displayed by FGPLOT. The user can move the cursor with the keyboard commands "R" (right) and "L" (left). The keyboard command "J" jumps 10 spaces in the direction last typed by the user. When the user types "M", the position of the cursor is returned in the ix and y parameters.

FGPTOV(savearea,curve) Plots a dynamic curve on the display that overlays the previously displayed curve.

BGAMMA(command) Exits from FORTRAN and loads the background GAMMA-11 program. BGAMMA is executed and it interprets the ASCII characters in the real

variable, command, as the first command. If command is blank (i.e., contains ASCII blanks), the background command table is displayed. If the characters form an illegal command, an error message is displayed, and typing a carriage return will return GAMMA-11 to the command table.

3.2.5 Linking Supplemental FORTRAN Subroutines With A User Program

There are three object files included in the supplemental FORTRAN support package. These are:

F4ROOT.OBJ
F4PLOT.OBJ
F4ADMN.OBJ

F4ROOT.OBJ should always be linked with your program whenever any of the supplemental support routines are referenced. F4PLOT.OBJ is linked when any curve plotting subroutines are used. F4ADMN.OBJ must be linked when any subroutines that reference GAMMA-11 patient studies are referenced within your program.

The following list shows subroutine calls that are referenced within the three files of the supplemental support package.

Object File	FORTRAN Supplemental Support Subroutines
F4ROOT	IBYTE, LBYTE, RSPI, ISPR, RDPI, RDPR, BGAMMA
F4PLOT	FGPLOT, FGPTOV, FGPICK
F4ADMN	FGADM1, FGADM2, FGCOM1, FGCOM2, IFGFRM, IGLSTR, IGLSTW

If the overlay feature of the RT-11 linker is used, F4ROOT should be linked to the root section of the program. F4ADMN and F4PLOT can be included in the overlays if desired (see following example).

1. If a program references only GAMMA-11 patient files, type:

```
.LINK PGM,F4ROOT,F4ADMN,FORLIB
```

2. If a program references only save area data and curves, type:

```
.LINK PGM,F4ROOT,F4PLOT,FORLIB
```

3. If a program references both GAMMA-11 patient files and save area data and curves, type:

```
.R LINK
PGM=PGM,F4ROOT,FORLIB/C
F4ADMN/O:1/C
F4PLOT/O:1
```

3.2.6 FORTRAN Example

The following program is an example of a FORTRAN program using the supplemental GAMMA-11 FORTRAN support.

```
C
C       READ A FRAME INTO ARRAY MAT
C
        DO 2 II=1,64,8
2       READ(1'N1) ((MAT(J,K),K=1,64),J=II,II+7)
C
C       CONVERT THE GAMMA DATA INTO FORTRAN FORMAT.
C       THEN CHECK EACH ELEMENT FOR NEW MAXIMUM
C
        DO 1 I=1,64
        DO 1 J=1,64
        MX=IBYTE(MAX(I,J))
        PT=IBYTE(MAT(I,J))
        IF(MX-PT .GE. 0) GOTO 1
C
C       IF NEW MAXIMUM, STORE COUNT AND TIME
C
        MAX(I,J)=MAT(I,J)
        TIM(I,J)=LBYTE(IJ)
1       CONTINUE
C
C       ELIMINATE COUNTS UNDER 5
C
        DO 12 I=1,64
        DO 12 J=1,64
        IF(IBYTE(MAX(I,J)) .LT. 5) TIM(I,J)=0
12      CONTINUE
C
C       RECONVERT INTO GAMMA FORMAT
C
        CALL FGADM2(ADMIN,P,B)
        N2=1
C
C       WRITE ALL BLOCKS PRECEDING DATA
C
        WRITE(2'N2) ADMIN
        DO 15 KK=2,MDOFF-1
        READ(1'KK)ADMIN
15      WRITE(2'N2) ADMIN
C
C       WRITE EITHER TIME OR COUNTS INTO FRAME
C
        WRITE(5,444)
444     FORMAT(' ', 'DO YOU WANT TIME OR INTENSITY FOR THE MATRIX?')
        WRITE (5,445)
445     FORMAT(' ','TYPE 1 FOR TIME, 2 FOR MAX. INTENSITY'//)
        READ(5,446) IP
446     FORMAT(I1)
        IF(IP-2 .GE. 0) GOTO 122
        DO 7 I=1,64,8
7       WRITE(2'N2)((TIM(J,K),K=1,64),J=I,I+7)
        STOP
122     DO 8 I=1,64,8
8       WRITE(2'N2) ((MAX(J,K),K=1,64),J=I,I+7)
        STOP
        END
```

```
C
C       FORTRAN SUPPORT EXAMPLE
C
C       ** FUNCTIONAL IMAGING PROGRAM **
C
C       THIS PROGRAM WILL STEP THROUGH A DYNAMIC STUDY LOOKING
C       FOR MAXIMUM VALUES AT EVERY MATRIX POSITION (I,J).
C       IT WILL RECORD THE TIME OF EACH MAXIMUM ALSO (TIM(64,64)).
C
        LOGICAL*1 MAT(64,64),TIM(64,64),MAX(64,64),ADMIN(512)
        INTEGER*2 N1,N2,NAME,NAME2,P(75),MX,PT,IP
        REAL*4 B(3,42),FM,IM
C
C       ZERO VARIABLES IN ARRAYS
C
        DATA FM/'FN I'/
        DATA IM/'MAGE'/
        DATA ADMIN/512*0/
        DATA P/75*0/,B/126*0./
        DO 333 I=1,64
        DO 334 J=1,64
        MAT(I,J)=0
        MAX(I,J)=0
334     TIM(I,J)=0
333     CONTINUE
C
C       INPUT FILE NAMES FROM KEYBOARD
C
        WRITE(5,222)
222     FORMAT(' ','INPUT PATIENT FILE AND NEW FILE NAME '//)
        CALL ASSIGN(1,NAME,-1,'RDO','NC')
        CALL ASSIGN(2,NAME2,-1,'NEW','NC')
C
C       DEFINE FILES FOR RANDO ACCESS I/O
C
        DEFINE FILE 1 (1000,256,U,N1)
        DEFINE FILE 2 (20,256,U,N2)
        N1=1
C
C       READ BLOCK 0 (RECORD 1) AND CONVERT FROM GAMMA FORMAT
C       TO FORTRAN FORMAT
C
        READ(1'N1) ADMIN
        CALL FGADM1(ADMIN,P,B)
        MDOFF=IFGFRM(P,1)
        N1=MDOFF
C
C       CONVERTR FRAMETERS FROM DYNAMIC TO STATIC
C
        P(1)=1
        P(14)=3
        P(20)=1
C
C       STORE 'FUNCTIONAL IMAGE' MESSAGE
C
        B(1,10)=FM
        B(2,10)=IM
        NFRMS=P(9)
        DO 1 IJ=1,NFRMS
```

CHAPTER 4

ASSEMBLING AND LINKING GAMMA-11

You can assemble and link GAMMA-11 by using the RT-11 MACRO and LINK commands. Indirect command files which contain all the commands required to assemble and link GAMMA-11 are included on the GAMMA-11 source media (DEC-11-MGAMA-C-EC, ED or ET).

4.1 ASSEMBLING GAMMA-11 USING INDIRECT COMMAND FILES

To assemble GAMMA-11, assign three logical devices and call two out of three indirect command files. The three indirect command files for assembling GAMMA-11 are

GMASMC.COM	Display-independent assemblies
GMASMV.COM	VSV01-dependent assemblies
GMASMS.COM	VT01-dependent assemblies

To assemble GAMMA-11 for the VSV01 color display, use command files GMASMC and GMASMV. To assemble GAMMA-11 for the VT01 display storage scope use command files GMASMC and GMASMS.

These indirect command files use three logical device assignments:

SRC	for the source file device (MACRO-11 input device)
OBJ	for the object file device (MACRO-11 output device)
LST	for the listing device (MACRO-11 listing device)

You must use the RT-11 ASSIGN command to assign physical devices to the logical devices before using the indirect command files. If you do not want the object files or the listing files, assign the null device handler (NL) to OBJ or LST.

NOTE

Each indirect command file generates about 3000 blocks of listings. Thus, if you assign an RK05 to LST, this disk becomes full if other files are also on it.

For example, the following RT-11 commands would be used to assemble GAMMA-11 for the VSV01 display with the source files on RK05 drive 0, the object files on RK05 drive 1, and the listing on the line printer:

```
.ASSIGN RK0 SRC
.ASSIGN RK1 OBJ
.ASSIGN LP LST
.@GMASMC
.@GMASMV
```

```
!************** GMASMC.COM ************************
!
!
!GAMMA-11 V2C DEVICE INDEPENDENT ASSEMBLIES
!
!
!****** MISC, SYSTEM SUMMARY, TRANSFER, DELETE ***
!
MACRO/LIST:LST:DATTIM/OBJ:OBJ:DATTIM/ALL:20. SRC:DATTIM
MACRO/LIST:LST:ACQDEV/OBJ:OBJ:ACQDEV/ALL:20. SRC:ACQDEV
MACRO/LIST:LST:GAMFIL/OBJ:OBJ:GAMFIL/ALL:20. SRC:GAMFIL
MACRO/LIST:LST:MEMMNG/OBJ:OBJ:MEMMNG/ALL:20. SRC:MEMMNG
MACRO/LIST:LST:SYSSUM/OBJ:OBJ:SYSSUM/ALL:20. SRC:(SYSSUM+GAMLIB/LIB)
MACRO/LIST:LST:DELETE/OBJ:OBJ:DELETE/ALL:20. SRC:(DELETE+GAMLIB/LIB)
MACRO/LIST:LST:TRNFER/OBJ:OBJ:TRNFER/ALL:40. SRC:(TRNFER+GAMLIB/LIB)
!
!************ DATA ACQUISITION  ********************
!
MACRO/LIST:LST:BACQCM/OBJ:OBJ:BACQCM/ALL:20. SRC:(ACQCMN+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:FACQCM/OBJ:OBJ:FACQCM/ALL:20. SRC:(FJDB+ACQCMN+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:EACQCM/OBJ:OBJ:EACQCM/ALL:20. SRC:(EFJDB+ACQCMN+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:DYNACQ/OBJ:OBJ:DYNACQ/ALL:20. SRC:(DYNACQ+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:EDYNAQ/OBJ:OBJ:EDYNAQ/ALL:20. SRC:(EFJDB+DYNACQ+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:STCACQ/OBJ:OBJ:STCACQ/ALL:20. SRC:(STCACQ+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:ESTCAQ/OBJ:OBJ:ESTCAQ/ALL:20. SRC:(EFJDB+STCACQ+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:LSTACQ/OBJ:OBJ:LSTACQ/ALL:20. SRC:(LSTACQ+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:FLSTAQ/OBJ:OBJ:FLSTAQ/ALL:20. SRC:(FJDB+LSTACQ+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:ELSTAQ/OBJ:OBJ:ELSTAQ/ALL:20. SRC:(EFJDB+LSTACQ+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:BACQSB/OBJ:OBJ:BACQSB/ALL:20. SRC:(ACQSB+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:FACQSB/OBJ:OBJ:FACQSB/ALL:20. SRC:(FJDB+ACQSB+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:EACQSB/OBJ:OBJ:EACQSB/ALL:20. SRC:(EFJDB+ACQSB+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:BACQST/OBJ:OBJ:BACQST/ALL:20. SRC:(ACQSTR+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:FACQST/OBJ:OBJ:FACQST/ALL:20. SRC:(FJDB+ACQSTR+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:EACQST/OBJ:OBJ:EACQST/ALL:20. SRC:(EFJDB+ACQSTR+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:BAQSET/OBJ:OBJ:BAQSET/ALL:90. SRC:(AQS1+AQS2+GSASET+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:FAQSET/OBJ:OBJ:FAQSET/ALL:90. SRC:(FJDB+AQS1+AQS2+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:EAQSET/OBJ:OBJ:EAQSET/ALL:90. SRC:(EFJDB+AQS1+AQS2+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:PREDEF/OBJ:OBJ:PREDEF/ALL:20. SRC:(PREDEF+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:GSAACQ/OBJ:OBJ:GSAACQ/ALL:20. SRC:(GSAACQ+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:PATMON/OBJ:OBJ:PATMON/ALL:20. SRC:(PATMON+APSECT+GAMLIB/LIB)
MACRO/LIST:LST:PADMIN/OBJ:OBJ:PADMIN/ALL:20. SRC:(SB+APSECT)
MACRO/LIST:LST:RWAVE/OBJ:OBJ:RWAVE/ALL:20. SRC:RWAVE
!
```

```
!************ DATA ANALYSIS **********************
!
MACRO/LIST:LST:FRMST1/OBJ:OBJ:FRMST1/ALL:20. SRC:(FRMST1+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:FRMST2/OBJ:OBJ:FRMST2/ALL:20. SRC:(FRMST2+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:ADMIN/OBJ:OBJ:ADMIN/ALL:20. SRC:(ADMIN+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:FLOOD/OBJ:OBJ:FLOOD/ALL:20. SRC:(FLOOD+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:INDEX/OBJ:OBJ:INDEX/ALL:20. SRC:(INDEX+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:DX/OBJ:OBJ:DX/ALL:20. SRC:(DX+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:ERROR/OBJ:OBJ:ERROR/ALL:20. SRC:(ERROR+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:COMMACRO/OBJ:OBJ:COMMACRO/ALL:20. SRC:(COMMAC+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:FLOLV2/OBJ:OBJ:FLOLV2/ALL:20. SRC:(FLOLV2+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:VMARK2/OBJ:OBJ:VMARK2/ALL:20. SRC:(VMARK2+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:ROILV2/OBJ:OBJ:ROILV2/ALL:20. SRC:(ROILV2+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:PPPLV2/OBJ:OBJ:PPPLV2/ALL:20. SRC:(PPPLV2+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:DATAR1/OBJ:OBJ:DATAR1/ALL:20. SRC:(DATAR1+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:JOY/OBJ:OBJ:JOY/ALL:20. SRC:(JOY+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:LIST/OBJ:OBJ:LIST/ALL:20. SRC:(LIST+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:FPMPEX/OBJ:OBJ:FPMPEX/ALL:20. SRC:FPMPEX
MACRO/LIST:LST:BUFSET/OBJ:OBJ:BUFSET/ALL:20. SRC:BUFSET
!
!************ BASIC SUPPORT **********************
!
MACRO/LIST:LST:GMBERR/OBJ:GMBERR/ALL:20. SRC:(ERRGAM)
!
!************ FORTRAN SUPPORT ********************
!
MACRO/LIST:LST:GMFERR/OBJ:GMFERR/ALL:20. SRC:(FORT+ERRGAM)
!
!*************** END ******************************



!************ GMASMV.COM ***********************
!
!
!GAMMA-11 V2C VSV01 DEPENDENT ASSEMBLIES
!
!
!************ BGAMMA ****************************
!
MACRO/LIST:LST:GAMRUM/OBJ:GAMRUM SRC:(GAMLIB/LIB+GAMRUM)
MACRO/LIST:LST:BGCOMD/OBJ:BGCOMD SRC:(GAMLIB/LIB+BGCOMD)
!
!************ DATA ANALYSIS **********************
!
MACRO/LIST:LST:DATARN/OBJ:OBJ:DATARN/ALL:20. SRC:(DATAR+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:GSAFRM/OBJ:OBJ:GSAFRM/ALL:20. SRC:(GSAFRM+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:VTDISP/OBJ:OBJ:VTDISP/ALL:20. SRC:(VTDISP+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:VTTEXT/OBJ:OBJ:VTTEXT/ALL:20. SRC:(VTTEXT+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:COLEDT/OBJ:OBJ:COLEDT/ALL:20. SRC:(COLEDT+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:COMND1/OBJ:OBJ:COMND1/ALL:20. SRC:(COMND1+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:IC/OBJ:OBJ:IC/ALL:20. SRC:(IC+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:PBSTOR/OBJ:OBJ:PBSTOR/ALL:20. SRC:(PBSTOR+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:PBMERG/OBJ:OBJ:PBMERG/ALL:20. SRC:(PBMERG+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:PBACK/OBJ:OBJ:PBACK/ALL:20. SRC:(PBACK+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:RDI/OBJ:OBJ:RDI/ALL:20. SRC:(RDI+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:PPP/OBJ:OBJ:PPP/ALL:20. SRC:(PPP+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:INIT/OBJ:OBJ:INIT/ALL:20. SRC:(INIT+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:SLICE/OBJ:OBJ:SLICE/ALL:20. SRC:(SLICE+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:GSATOL/OBJ:OBJ:GSATOL/ALL:20. SRC:(GSATOL+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:INITRI/OBJ:OBJ:INITRI/ALL:20. SRC:(INITRI+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:COMND2/OBJ:OBJ:COMND2/ALL:20. SRC:(COMND2+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:DUAL/OBJ:OBJ:DUAL/ALL:20. SRC:(DUAL+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:DISCMD/OBJ:OBJ:DISCMD/ALL:20. SRC:(DISCMD+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:DATAR1/OBJ:OBJ:DATAR1/ALL:20. SRC:(DATAR1+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:DISPAT/OBJ:OBJ:DISPAT/ALL:20. SRC:(DISPAT+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:VTWRIT/OBJ:OBJ:VTWRIT/ALL:20. SRC:(VTWRIT+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:NRCTBL/OBJ:OBJ:NRCTBL/ALL:20. SRC:(NRCTBL+CSECT+GAMLIB/LIB)
!
```

```
!************ BASIC SUPPORT **********************
!
MACRO/LIST:LST:GMBAS1/OBJ:GMBAS1/ALL:20. SRC:(F4BAS+RTFB+IOTBL+CSECT1)
MACRO/LIST:LST:GMBAS2/OBJ:GMBAS2/ALL:20. SRC:(F4BAS2+CSECT1)
MACRO/LIST:LST:GMBAS3/OBJ:GMBAS3/ALL:20. SRC:(PLOT+CSECT1)
MACRO/LIST:LST:GAMCLI/OBJ:GAMCLI/ALL:20. SRC:GAMCLI
!
!************ FORTRAN SUPPORT ********************
!
MACRO/LIST:LST:GMFOR1/OBJ:GMFOR1/ALL:20. SRC:(FORT+F4BAS+RTFB+IOTBL+CSECT1)
MACRO/LIST:LST:GMFOR2/OBJ:GMFOR2/ALL:20. SRC:(FORT+F4BAS2+CSECT1)
MACRO/LIST:LST:GMFOR3/OBJ:GMFOR3/ALL:20. SRC:(FORT+PLOT+CSECT1)
MACRO/LIST:LST:F4ADMN/OBJ:F4ADMN/ALL:20. SRC:(COLOR+F4ADM)
MACRO/LIST:LST:F4ROOT/OBJ:F4ROOT/ALL:20. SRC:(COLOR+ROOT+CSCTV1)
MACRO/LIST:LST:F4PLOT/OBJ:F4PLOT/ALL:20. SRC:(COLOR+PLOTV1+CSCTV1)
!
!****************** END ***************************



!************** GMASMS.COM ***********************
!
!
!GAMMA-11 V2C VT01 DEPENDENT ASSEMBLIES
!
!
!************ BGAMMA *****************************
!
MACRO/LIST:LST:GAMRMS/OBJ:GAMRMS SRC:(VT01+GAMRUM+GAMLIB/LIB)
MACRO/LIST:LST:BGCMDS/OBJ:BGCMDS SRC:(VT01+BGCMD+GAMLIB/LIB)
!
!************ DATA ANALYSIS **********************
!
MACRO/LIST:LST:DATARS/OBJ:OBJ:DATARS/ALL:20. SRC:(VT01+DATAR+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:MDIS/OBJ:OBJ:MDIS/ALL:20. SRC:(MDIS+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:CMND1S/OBJ:OBJ:CMND1S/ALL:20. SRC:(VT01+CDMND1+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:ICS/OBJ:OBJ:ICS/ALL:20. SRC:(VT01+IC+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:RDIS/OBJ:OBJ:RDIS/ALL:20. SRC:(VT01+RDI+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:PPPS/OBJ:OBJ:PPPS/ALL:20. SRC:(PPP+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:INITS/OBJ:OBJ:INITS/ALL:20. SRC:(VT01+INIT+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:SLICE/OBJ:OBJ:SLICES/ALL:20. SRC:(VT01+SLICE+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:INTRIS/OBJ:OBJ:INTRIS/ALL:20. SRC:(VT01+INITRI+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:CMND2S/OBJ:OBJ:CMND2S/ALL:20. SRC:(VT01+CDMND2+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:DUALS/OBJ:OBJ:DUALS/ALL:20. SRC:(VT01+DUAL+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:DSCMDS/OBJ:OBJ:DSCMDS/ALL:20. SRC:(VT01+DISCMD+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:DSPATS/OBJ:OBJ:DSPATS/ALL:20. SRC:(VT01+DISPAT+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:VTWRTS/OBJ:OBJ:VTWRTS/ALL:20. SRC:(VT01+VTWRIT+CSECT+GAMLIB/LIB)
MACRO/LIST:LST:NRTBLS/OBJ:OBJ:NRTBLS/ALL:20. SRC:(VT01+NRCTBL+CSECT+GAMLIB/LIB)
!
!************ BASIC SUPPORT **********************
!
MACRO/LIST:LST:GMBVT1/OBJ:GMBAS1.OVT/ALL:20. SRC:(VT01+F4BAS+RTFB+IOTBL+CSECT1)
MACRO/LIST:LST:GMBVT2/OBJ:GMBAS2.OVT/ALL:20. SRC:(VT01+F4BAS2+CSECT1)
MACRO/LIST:LST:GMBVT3/OBJ:GMBAS3.OVT/ALL:20. SRC:(VT01+PLOT+CSECT1)
MACRO/LIST:LST:GAMCVT/OBJ:GAMCLI.OVT/ALL:20. SRC:(VT01+GAMCLI)
!
!************ FORTRAN SUPPORT ********************
!
MACRO/LIST:LST:GMFVT1/OBJ:GMFOR1.OVT/ALL:20. SRC:(VT01+FORT+F4BAS+RTFB+IOTBL+CSECT1)
MACRO/LIST:LST:GMFVT2/OBJ:GMFOR2.OVT/ALL:20. SRC:(VT01+FORT+F4BAS2+CSECT1)
MACRO/LIST:LST:GMFVT3/OBJ:GMFOR3.OVT/ALL:20. SRC:(VT01+FORT+PLOT+CSECT1)
MACRO/LIST:LST:F4ADVT/OBJ:F4ADMN.OVT/ALL:20. SRC:(VT01+F4ADM)
MACRO/LIST:LST:F4RTVT/OBJ:F4ROOT.OVT/ALL:20. SRC:(VT01+ROOT+CSCTV1)
MACRO/LIST:LST:F4PLVT/OBJ:F4PLOT.OVT/ALL:20. SRC:(VT01+PLOTV1+CSCTV1)
!
!****************** END  ***************************
```

4.2 LINKING GAMMA-11 USING INDIRECT COMMAND FILES

There are four indirect command files for linking GAMMA-11:

GMLNKC.COM	Display-independent links
GMLNKV.COM	VSV01-dependent links
GMLNKS.COM	VT01-dependent links
GMLNKB.COM	BASIC/RT-11 links (with GAMMA-11 subroutines)

To link GAMMA-11 for the VSV01 color display, use indirect command files GMLNKC and GMLNKV. To link GAMMA-11 for the VT01 display storage scope use indirect command files GMLNKC and GMLNKS.

These command files use four logical device assignments:

OBJ	for the object file device (LINK-11 input device)
EXE	for the .SAV and .REL files (LINK-11 output device)
MAP	for the link map device (LINK-11 map device)
BAS	for BASIC/RT-11 object files

BAS is used only by GMLNKB.COM.

Use the RT-11 ASSIGN command to assign physical devices to the logical devices. If you do not want output files or maps, assign the null device handler (NL) to EXE or MAP.

For example, the following RT-11 commands would be used to link GAMMA-11 for the VSV01 display with the object files on RK05 drive 1, the .SAV and .REL files on RK05 drive 1, and no link map.

```
.ASSIGN RK1 OBJ
.ASSIGN RK1 EXE
.ASSIGN NL MAP
.@GMLNKC
.@GMLNKV
```

```
!
!
!GAMMA-11 V2C DISPLAY INDEPENDENT LINKS
!
!
LINK/MAP:MAP:DATTIM/WIDE/EXE:SAV:DATTIM OBJ:DATTIM
LINK/MAP:MAP:SYSSUM/WIDE/EXE:SAV:SYSSUM OBJ:SYSSUM
LINK/MAP:MAP:DELETE/WIDE/EXE:SAV:DELETE OBJ:(DELETE,INDEX)
LINK/MAP:MAP:TRNFER/WIDE/EXE:SAV:TRNFER/PROMPT -
OBJ:TRNFER
OBJ:GAMFIL/O:1
OBJ:INDEX/O:1
//
LINK/MAP:MAP:DATACQ/WIDE/EXE:DATACQ/PROMPT -
OBJ:(BACQCM,GAMDEV)
OBJ:DYNACQ/O:1
OBJ:STCACQ,LSTACQ/O:1
OBJ:PATMON/O:2
OBJ:BACQSB/O:2
OBJ:GAMFIL/O:2
OBJ:BACQST/O:3
OBJ:PREDEF/O:3
OBJ:BAQSET/O:3
OBJ:PADMIN/O:4
OBJ:GSAACQ,RWAVE/O:5
//
```

```
LINK/MAP:MAP:FGAMMA/WIDE/EXE:SAV:FGAMMA/PROMPT/FORE -
OBJ:(FACQCM,GAMDEV)
OBJ:DYNACQ/O:1
OBJ:STCACQ/O:1
OBJ:FLSTAQ/O:1
OBJ:FACQSB/O:2
OBJ:GAMFIL/O:2
OBJ:FACQST/O:3
OBJ:PREDEF/O:3
OBJ:FAQSET/O:3
OBJ:PADMIN/O:4
//
LINK/MAP:MAP:EGAMMA/WIDE/EXE:SAV:EGAMMA/PROMPT/FORE -
OBJ:(EACQCM,GAMDEV)
OBJ:EACQSB/O:1
OBJ:GAMFIL/O:1
OBJ:EACQST/O:2
OBJ:PREDEF/O:2
OBJ:EAQSET/O:2
OBJ:EDYNAQ/O:3
OBJ:ESTCAQ/O:3
OBJ:ELSTAQ,RWAVE/O:3
OBJ:PADMIN,MEMMNG/O:4
//
!
!
!GAMMA-11 V2C VSV01 DEPENDENT LINKS
!
!
LINK/MAP:MAP:BGAMMA/WIDE/EXE:SAV:BGAMMA/PROMPT -
OBJ:(GAMRUM,GAMDEV,MEMMNG)
OBJ:BGCOMD/O:1
//
LINK/MAP:MAP:DATANL/WIDE/EXE:SAV:DATANL/PROMPT -
OBJ:DATARN
OBJ:FRMST1/O:1/C
OBJ:FRMST2/O:1/C
OBJ:GSAFRM/O:1/C
OBJ:VTDISP/O:1/C
OBJ:VTTEXT/O:1/C
OBJ:COLEDT/O:1/C
OBJ:ADMIN/O:1/C
OBJ:COMND1/O:1/C
OBJ:IC/O:1/C
OBJ:PBMERG/O:1/C
OBJ:PBSTOR/O:1/C
OBJ:PBACK/O:1/C
OBJ:ROI/O:1/C
OBJ:FLOOD/O:1/C
OBJ:PPP/O:1/C
OBJ:INDEX,INIT,DX/O:1/C
OBJ:ERROR/O:1/C
OBJ:COMMAC/O:1/C
OBJ:SLICE/O:1/C
OBJ:GSATOL/O:2/C
OBJ:FLOLV2/O:2/C
OBJ:INITRI/O:2/C
OBJ:VMARK2/O:2/C
OBJ:PPPLV2/O:2/C
OBJ:ROILV2/O:2/C
OBJ:COMND2/O:2/C
```

```
OBJ:DUAL/O:2/C
OBJ:DISCMD/O:2/C
OBJ:BUFSET/O:2/C
OBJ:DATAR1/O:3/C
OBJ:DISPAT/O:3/C
OBJ:JOY/O:3/C
OBJ:VTWRIT/O:3/C
OBJ:FPMPEX/O:3/C
OBJ:NRCTBL/O:3/C
OBJ:LIST/O:3
//
!
!
!GAMMA-11 V2C VT01 DEPENDENT LINKS
!
!
LINK/MAP:MAP:BGAMMS.MAP/WIDE/EXE:SAV:BGAMMA.VT1/PROMPT -
OBJ:(GAMRMS,GAMDEV)
OBJ:BGCMDS/O:1
//
LINK/MAP:MAP:DATNLS.MAP/WIDE/EXE:SAV:DATNLS/PROMPT -
OBJ:DATARS
OBJ:FRMST1/O:1/C
OBJ:FRMST2/O:1/C
OBJ:ADMIN/O:1/C
OBJ:CMND1S/O:1/C
OBJ:ICS/O:1/C
OBJ:ROIS/O:1/C
OBJ:FLOOD/O:1/C
OBJ:PPPS/O:1/C
OBJ:INDEX,INITS,DX/O:1/C
OBJ:ERROR/O:1/C
OBJ:COMMAC/O:1/C
OBJ:SLICES/O:1/C
OBJ:FLDLV2/O:2/C
OBJ:INTRIS/O:2/C
OBJ:VMARK2/O:2/C
OBJ:PPPLV2/O:2/C
OBJ:ROILV2/O:2/C
OBJ:CMND2S/O:2/C
OBJ:DUALS/O:2/C
OBJ:DSCMDS/O:2/C
OBJ:BUFSET/O:2/C
OBJ:DATR1S/O:3/C
OBJ:DSPATS/O:3/C
OBJ:JOY/O:3/C
OBJ:VTWRTS/O:3/C
OBJ:FPMPEX/O:3/C
OBJ:NRTBLS/O:3/C
OBJ:LIST/O:3
//
```

```
!****** LINK VSV01 BASIC W/O EIS *****
!
R LINK
SAV:BASIC,MAP:BASIC/W=/B:700//
BAS:BSPTRS,BSPAT,BSROS
OBJ:GAMCLI
OBJ:GMBAS1
BAS:BSCLLB
BAS:SUIOPR
BAS:SUO1ID,BSOTOS,BSOT1S/O:1
BAS:SUIOID,SUIOPO,BSPRO,SUR1ID,BSR1S/O:1
BAS:SUXID,BSX0A,BSX0B/O:2
BAS:SUX2ID,BSX2/O:2
BAS:SUEID,BSE0/O:2
BAS:SUE1ID,BSE1,BSSUB,BSRSQ/O:2
BAS:SUOTCM/O:2
BAS:BSERR,BSERML/O:2
BAS:BSX1A/O:3
BAS:BSX1B/O:3
BAS:BSPR1/O:3
BAS:BSCLS/O:3
BAS:BSKEYS/O:3
BAS:BSCMP/O:3
OBJ:GMBAS2/O:4
OBJ:GMBAS3/O:4
OBJ:GMBERR/O:4
BAS:SUIMP,BSFUNC,SUOPT/O:5
//
!
!****** LINK VSV01 BASIC WITH EIS *****
!
SAV:BASICE,MAP:BASICE/W=/B:700//
BAS:BSPTRS,BSPAT,BSROS
OBJ:GAMCLI
OBJ:GMBAS1
BAS:BSCLLB
BAS:SUIOPR
BAS:SUO1ID,BSOTOS.EIS,BSOT1S.EIS/O:1
BAS:SUIOID,SUIOPO,BSPRO,SUR1ID,BSR1S/O:1
BAS:SUXID,BSX0A,BSX0B/O:2
BAS:SUX2ID,BSX2/O:2
BAS:SUEID,BSE0/O:2
BAS:SUE1ID,BSE1,BSSUB,BSRSQ/O:2
BAS:SUOTCM/O:2
BAS:BSERR,BSERML/O:2
BAS:BSX1A/O:3
BAS:BSX1B/O:3
BAS:BSPR1/O:3
BAS:BSCLS/O:3
BAS:BSKEYS/O:3
BAS:BSCMP/O:3
OBJ:GMBAS2/O:4
OBJ:GMBAS3/O:4
OBJ:GMBERR/O:4
BAS:SUIMP,BSFUNC,SUOPT/O:5
//
```

```
!
!****** LINK VT01 BASIC W/O EIS *****
!
SAV:BASIC.VT1,MAP:BASICS/W=/B:700//
BAS:BSPTRS,BSPAT,BSROS
OBJ:GAMCLI.OVT
OBJ:GMBAS1.OVT
BAS:BSCLLB
BAS:SUIOPR
BAS:SUO1ID,BSOT0S,BSOT1S/O:1
BAS:SUIOID,SUIOPO,BSPRO,SUR1ID,BSR1S/O:1
BAS:SUXID,BSX0A,BSX0B/O:2
BAS:SUX2ID,BSX2/O:2
BAS:SUEID,BSE0/O:2
BAS:SUE1ID,BSE1,BSSUB,BSRSQ/O:2
BAS:SUDTCM/O:2
BAS:BSERR,BSERML/O:2
BAS:BSX1A/O:3
BAS:BSX1B/O:3
BAS:BSPR1/O:3
BAS:BSCLS/O:3
BAS:BSKEYS/O:3
BAS:BSCMP/O:3
OBJ:GMBAS2.OVT/O:4
OBJ:GMBAS3.OVT/O:4
OBJ:GMBERR/O:4
BAS:SUIMP,BSFUNC,SUOPT/O:5
//
!
!****** LINK VT01 BASIC WITH EIS *****
!
SAV:BASICE.VT1,MAP:BASCES/W=/B:700//
BAS:BSPTRS,BSPAT,BSROS
OBJ:GAMCLI.OVT
OBJ:GMBAS1.OVT
BAS:BSCLLB
BAS:SUIOPR
BAS:SUO1ID,BSOT0S.EIS,BSOT1S.EIS/O:1
BAS:SUIOID,SUIOPO,BSPRO,SUR1ID,BSR1S/O:1
BAS:SUXID,BSX0A,BSX0B/O:2
BAS:SUX2ID,BSX2/O:2
BAS:SUEID,BSE0/O:2
BAS:SUE1ID,BSE1,BSSUB,BSRSQ/O:2
BAS:SUDTCM/O:2
BAS:BSERR,BSERML/O:2
BAS:BSX1A/O:3
BAS:BSX1B/O:3
BAS:BSPR1/O:3
BAS:BSCLS/O:3
BAS:BSKEYS/O:3
BAS:BSCMP/O:3
OBJ:GMBAS2.OVT/O:4
OBJ:GMBAS3.OVT/O:4
OBJ:GMBERR/O:4
BAS:SUIMP,BSFUNC,SUOPT/O:5
//
^C
!
!
```

APPENDIX A

BASIC/RT-11 LANGUAGE SUMMARY

This appendix lists the BASIC/RT-11 commands, functions, statements, and error messages. For more detail, see the BASIC-11 Language Reference Manual (DEC-11-LIBBB-A-D) and the BASIC-11/RT-11 User's Guide (DEC-11-LIBUA-A-D).

For the differences between version 1B and Version 2 of BASIC, see the BASIC-11/RT-11 Installation Guide (DEC-11-LIBTA-A-D).

A.1 BASIC/RT-11 STATEMENTS

CALL routine name [[(argument list)]]

Calls assembly language routines from a BASIC program.

CHAIN string [[LINE expression]]

Terminates execution of the program, loads the program specified by string, and begins execution at the lowest line number or at the line number specified by expression. The string is a file specification.

CLOSE[[[#]expr1,[#]expr2,[#]expr3, ...]]

Closes the file(s) associated with the channel number(s) and virtual file channel number(s) specified. If no channel number is specified, closes all open files.

COMMON list

Preserves values and names of specified variables and arrays when the CHAIN statement is executed. Both string and arithmetic variables and arrays can be passed. The statement also dimensions the specified arrays. List is in the general format:

var1[[(expr[[,expr]])]][[,var2[[(expr[[,expr]])]],...]]

DATA list

Used in conjunction with READ to input listed data into an executing program. Can contain any mixture of strings and numbers. Items must be separated by commas.

DEF FNletter[[($|%)]](var1 [[,var2,...,var5]])=expression

Defines a user function. Letter may be any single letter A through Z.

DIM list

Reserves space in memory for arrays according to the subscript(s) specified after the variable name. List is in the general format:

var1(expr [[,expr]]) [[,var2(expr [[,expr]]),...]]

DIM #integer1,variable(integer2 ,integer3) =integer4

Dimensions the virtual array file associated with the channel number specified by integer1. Integer4 specifies the string size for string virtual arrays.

END

Optional. Placed at the physical end of the program to terminate execution.

FOR var=expr1 TO expr2 [[STEP expr3]]

Sets up a loop to be executed the specified number of times.

GOSUB line number

Unconditionally transfers control to specified line of subroutine.

GO TO line number

Unconditionally transfers control to specified line number.

IF relational expression {THEN statement / THEN line number / GO TO line number}

Conditionally executes the specified statement or transfers control to specified line number. When the condition is not true and a statement is specified, execution continues at the next sequential statement. The expressions and the relational operator must all be string or all be numeric.

IF END #expr {THEN statement / THEN line number / GO TO line number}

Tests for end-of-file condition of input sequential file associated with channel number specified by expression.

INPUT [[#expr,]] variable1 [[,variable2,...]]

Inputs data from your terminal or from the file associated with the channel number specified by expression. Variables may be arithmetic or string.

KILL string

Deletes file specified by string.

[[LET]] variable=expression

Assigns value of expression to the specified variable. Variable and expression must be of the same type, either numeric or string.

LINPUT [[#expr,]] string var1 [[,string var2,...]]

Inputs string data from the terminal or from the file associated with channel number specified by expression. Variables can only be string variables.

NAME string1 TO string2

Renames file specified by string1 to name specified by string2.

NEXT variable

Placed at end of FOR loop to return control to FOR statement.

ON expression GOSUB line number1 [[,line number2,line number3,...]]

Conditionally transfers control to subroutine at one line number specified in list. Value of expression determines the line number to which control is transferred.

ON expression GO TO line number1 [[,line number2,line number3,...]]

Conditionally transfers control to one line number in the list. Value of expression determines the line number to which control is transferred.

ON expression THEN line number1 [[,line number2,...]]

Equivalent to ON GO TO.

OPEN string [{FOR INPUT / FOR OUTPUT}] AS FILE [[#]] expr1 [[DOUBLE BUF]] [[,RECORDSIZE expr2]] [[,MODE expr3]] [[,FILESIZE expr4]]

Opens a file specified by string for input or output as specified (assumes input if neither specified) and associates file with the channel number specified by expr1. String is a file specification.

OVERLAY string [[LINE expression]]

Overlays or merges the program currently in memory with the program in the file specified by string, and when overlay is completed, transfers control to either the next sequential BASIC line number or the line number specified by expression. String is a file specification.

PRINT [[#expr,]] [[list]]

Prints items in list on the terminal or to the file associated with channel number specified by expression. List can consist of string and arithmetic expressions and the TAB function. Items can be separated by either commas or semicolons.

PRINT [[#expr,]] USING string, list

Prints items in list on the terminal or to the file associated with channel number specified by expr in the format determined by string. List can consist of string and arithmetic expressions. Items can be separated by either commas or semicolons.

RANDOMIZE

Causes the random number generator (RND function) to produce different random numbers.

READ variable1 [[,variable2,...]]

Assigns values listed in DATA statements to specified variables. Variables may be string or numeric.

REM comment

No effect on execution of program. Contains explanatory comments about the BASIC program.

RESET[[#expr]]

Equivalent to RESTORE.

RESTORE[[#expr]]

Resets either the data pointer or, when specified, the input file associated with the specified channel number to the beginning.

RETURN

Terminates a subroutine and returns control to the statement following the last executed GOSUB statement.

STOP

Terminates execution of the program. Placed at logical end(s) of the program.

A.2 SUMMARY OF BASIC/RT-11 FUNCTIONS

Arithmetic Functions

ABS(expr)

Returns the absolute value of the expression.

ATN(expr)

Returns the arctangent of the expression as an angle in radians in the range + or - pi/2.

COS(expr)

Returns the cosine of the angle specified by the expression in radians.

EXP(expr)

Returns the value of e raised to the power (expr) where e is (approximately) 2.71828.

INT(expr)

Returns the greatest integer less than or equal to the expression (expr). (Truncation of decimal values.)

LOG(expr)

Returns the natural logarithm of the expression (expr).

LOG10(expr)

Returns the base 10 logarithm of the expression (expr).

PI

Returns the value of pi (3.141593).

RND [[(expr)]]

Returns a random number between 0 and 1.

SGN(expr)

Returns a value indicating the sign of expression (expr).

SIN(expr)

Returns the sine of the angle specified by expression (expr) in radians.

SQR(expr)

Returns the square root of the expression (expr).

TAB(expr)

Causes the terminal to tab to column number specified by the expression (expr) (valid only in PRINT statements).

String Functions

ASC(string)

Returns as a decimal number the 8-bit internal code (ASCII value) for the 1-character string expression (string).

BIN(string)

Converts a string expression (string) containing a binary number to a decimal value. Blanks are ignored.

CHR$(expr)

Generates a 1-character string whose ASCII value is the low-order 8 bits of the integer value of the expression (expr).

CLK$

Returns the time as a string in the form hh:mm:ss (for example 12:30:15).

DAT$

Returns the date as a string in the form dd-mon-yr (for example 07-FEB-75).

LEN(string)

Returns the number of characters in the string (string).

OCT(string)

Converts a string expression (string) containing an octal number to a decimal value. Blanks are ignored.

POS(string1,string2,expr)

Searches for and returns the position of the first occurrence of string2 in string1. The search starts at the character position specified by expression (expr).

SEG$(string,expr1,expr2)

Returns the string of characters in position specified by expression1 through the position specified by expression2.

STR$(expr)

Returns the string which represents the numeric value of the expression.

TRM$(string)

Returns string without trailing blanks.

VAL(string)

Returns the value of the decimal number contained in the string.

System Functions

ABORT(expr)

Deletes the program and changes the program name to NONAME if the expression is equal to 1. The ABORT function is equivalent to an END statement if the expression is equal to 0.

CTRLC

Enables the BASIC program to be interrupted with a CTRL/C.

RCTRLC

Disables the CTRL/C interrupt. While the RCTRLC function is in effect, the BASIC program cannot be interrupted.

RCTRLO

Ensures that BASIC program output is printed even if a CTRL/O is in effect.

SYS(expr1[,expr2])

Performs system dependent functions determined by expr1 and expr2. See the BASIC-11/RT-11 User's Guide (DEC-11-LIBTA-A-D).

TTYSET (255%,expr)

Specifies the right margin of the terminal as the value of expr-1. If expr equals 0, BASIC does not change the previous margin.

A.3 SUMMARY OF BASIC/RT-11 COMMANDS

APPEND [[file specification]]

Merges the program in your area in memory with the program specified by the file specification.

CLEAR

Initializes all variables to 0 and all string variables to nulls and deletes arrays.

COMPILE [[file specification]]

Saves a compiled version of the program.

DEL line specification [[,line specification,...]]

Deletes specified lines.

LENGTH

Prints on your terminal the size of the program in memory and the size of the remaining free memory.

LIST[[NH]] [[line specification1,line specification2,...]]

Prints on the terminal the specified line(s) of the program currently in memory. NH suppresses the printing of the header line and is optional.

NEW [[program name]]

Erases your storage area and sets the current program name to the one specified.

OLD [[file specification]]

Erases your storage area and inputs the program from the specified file.

RENAME program name

Changes the current program name to the one specified.

REPLACE [[file specification]]

Replaces the specified file with the current program.

RESEQ [[[[new line number]],[[old line number1]] [[-old line number2]],[[increment]]]]

Resequences program as specified.

RUN[[NH]]

Executes the program in memory. NH suppresses the printing of the header line and is optional.

RUN[[NH]]file specification

Erases your storage area, inputs the program from the specified file, and then executes the program. Does not print header line in any case.

SAVE [[file specification]]

Outputs the program in memory to the specified file.

SCR

Erases your storage area and changes the program name to NONAME.

SUB line numberxstringlxstring2[[xinteger]]

Substitutes the integer occurrence of string1 with string2 on line specified. x is a delimiter and can be any character such as @.

UNSAVE file specification

Deletes specified file.

Key Commands

CTRL/C

Interrupts execution of a command or program and causes BASIC to print the READY message. See your BASIC-11 User's Guide for more information about CTRL/C.

CTRL/O

Causes all further terminal output to be discarded. Printing resumes if an INPUT statement is encountered, another CTRL/O is typed, or the program is terminated.

CTRL/Q

Continues output to the terminal; cancels effect of CTRL/S.

CTRL/S

Temporarily suspends all output to terminal until CTRL/Q is typed; allows alphanumeric display terminals to be read or photographed before data is moved off screen.

CTRL/U

Deletes the entire current input line (provided the RETURN key has not been typed).

DELETE

Deletes the last character typed.

A.4 BASIC/RT-11 ERROR MESSAGES

?ARGUMENT ERROR (?ARG)

Arguments in a function do not match the arguments defined for the function, in number, range, or type. Ensure that there are the correct number of arguments, that their values are in the correct range, and that they are the correct type.

?ARRAYS TOO LARGE (?ATL)

Not enough memory is available for the arrays specified in the DIM statements. Reduce the size of the arrays or reduce the size of the program.

?BAD DATA READ (?BDR)

Data item input from a DATA statement or from a file is the wrong data type. Ensure that the DATA statement or the file contains the same data type as specified in the READ or INPUT # statement.

?BAD DATA - RETYPE FROM ERROR (?BRT)

Nonfatal. Item entered in response to an INPUT or INPUT #0 statement is the wrong data type. Retype item and program will continue.

?BAD LOG (?BLG)

Nonfatal. Expression in LOG or LOG10 function is 0 or negative. The function returns 0 and BASIC continues execution of the program.

?BUFFER STORAGE OVERFLOW (?BSO)

Not enough room available for file buffer in your area. Reduce program size.

?CHANNEL ALREADY OPEN (?CAO)

OPEN statement specifies a channel that is already associated with an open file. Ensure that OPEN statements specify correct channel numbers and that files that should be closed are closed.

?CHANNEL I/O ERROR (?CIE)

Accessing data in a file produces an error. Ensure that your peripheral devices and their storage media are working correctly. One possible cause is that the file accessed has 0 length.

?CHANNEL NOT OPEN (?CNO)

A PRINT #, PRINT # USING, INPUT #, IF END #, or CLOSE statement, or a reference to a virtual array file specifies a channel number not associated with an open file. Check that the OPEN statement has been executed and that it specifies the same channel number as the program line with the error.

?CHECKSUM ERROR IN COMPILED PROGRAM (?CCP)

File produced by the COMPILE command contains a format error. Use a copy of the program created by a SAVE or REPLACE command.

?COMMON OUT OF ORDER (?COO)

Variables and arrays in a COMMON statement are not listed in the same order as those in a previous segment. Ensure that all segments have equivalent COMMON statements.

?CONTROL VARIABLE OUT OF RANGE (?CVO)

Expression in an ON GOTO or ON GOSUB statement is 0 or negative or has a value greater than the number of line numbers listed. Ensure that expression has a value in the correct range.

?DIVISION BY ZERO (?DV0)

Nonfatal. An expression includes a division by 0. BASIC substitutes a value of 0 for that operation and continues execution of the program.

?END NOT LAST (?ENL)

END statement is not the highest numbered program line. This error message is printed when the END statement is executed. Ensure that there is only one END statement in program and that it has the highest line number.

?EXCESS INPUT IGNORED (?EII)

Nonfatal. There are more data items than required by an INPUT or INPUT #0 statement. BASIC ignores the excess items and continues execution of the program. Ensure that data items did not contain an unintended comma (e.g., 1,430 instead of 1.430).

?EXPONENTIATION ERROR (?ERR)

Nonfatal. An expression includes the operation of raising a negative value to a nonintegral power (e.g., (-1)^.5). This would produce a complex number, which cannot be represented in

BASIC. This message is also produced when a negative value is raised to an integral value that has an absolute value greater than 255 (e.g., (-1)^256). In both cases, BASIC substitutes a value of 0 for the operation and continues execution.

?EXPRESSION TOO COMPLEX (?ETC)

An expression is too complex for BASIC to evaluate in the area it uses for calculations (called the stack). This condition is usually caused by including user-defined functions or nested functions in an expression. The degree of complexity that causes this error varies according to the amount of space available in the stack at the time. Breaking the statement up into several statements containing simpler expressions may eliminate the error.

?FILE NOT FOUND (?FNF)

BASIC cannot find the specified file. Ensure that the file specification was typed correctly and that the file exists.

?FILE TOO SHORT (?FTS)

The file is too small to contain the output. If the error occurs in a data file, specify a larger FILESIZE. If the error occurs in a program file, delete unused files with the UNSAVE command and then retry.

?FLOATING OVERFLOW (?FOV)

Nonfatal. The absolute value of the result of a computation is greater than the largest number that can be stored by BASIC (approximately 10^38). BASIC substitutes a value of 0 for the operation and continues execution of the program.

?FLOATING UNDERFLOW (?FUN)

Nonfatal. The absolute value of the result of a computation is smaller than the smallest number that BASIC can store (approximately 10^(-38) . BASIC substitutes a value of 0 for operation and continues execution of the program.

?FOR WITHOUT NEXT (?FWN)

The program contains a FOR statement without a corresponding NEXT statement to terminate the loop. Ensure that each loop in the program is terminated with a NEXT statement.

?FUNCTION ALREADY DEFINED (?FAD)

The user-defined function is previously defined. Ensure that each function is defined only once and has a unique name.

?ILLEGAL CHANNEL NUMBER (?ICN)

The channel specified is not in the range allowed or the IF END statement specifies a file on a terminal. See your BASIC-11 user's guide for information about the range of valid channel numbers.

?ILLEGAL DIM (?IDM)

A subscript in a DIM or COMMON statement is not an integer, an array is dimensioned more than once, or an array has more than two dimensions. Ensure that an array specification is in the correct format and appears only once in the COMMON and DIM statements in the program.

?ILLEGAL END OF FILE IN COMPILED PROGRAM (?IEF)

File produced by the COMPILE command contains a format error. Use a copy of the program created by a SAVE or REPLACE command.

?ILLEGAL FILE LENGTH (?IFL)

The FILESIZE specified was less than -1.

?ILLEGAL FILE SPECIFICATION (?IFS)

The file specification is invalid. See your BASIC-11 user's guide for information on the format of a file specification.

?ILLEGAL IN IMMEDIATE MODE (?IIM)

The INPUT or INPUT # statement cannot be entered in immediate mode. Enter the statement in a program line (followed with a STOP statement) and execute the statement with an immediate mode GO TO statement.

?ILLEGAL I/O DIRECTION (?IID)

Statement attempts to write to an input file or read an output file. Ensure that the channel number specified specifies the correct file. If the statement assigns a value to an element of a virtual array file, ensure that the file's OPEN statement does not specify "FOR INPUT."

?INCONSISTENT NUMBER OF SUBSCRIPTS (?INS)

The array is dimensioned with one subscript and referenced by two, or vice versa. Ensure that the DIM statement and array references are consistent.

?INPUT STRING ERROR (?ISE)

Nonfatal. A string entered in response to an INPUT statement begins with a quotation mark but is not terminated by the appropriate end quotation mark. BASIC assigns to the string all the characters between the initial quote and the line terminator and continues execution of the program.

?INTEGER OVERFLOW (?IOV)

An integer variable is assigned a value greater than 32767 or less than -32768 or an integer expression produces a result which exceeds this range. Change the variable or expression to a floating point format.

?LINE TOO LONG (?LTL)

The line entered is longer than BASIC allows; the line is ignored. If this message occurs when BASIC is reading a program from a file, BASIC stops reading the file. A possible cause is that you entered a line near the maximum size with no spaces, but when you save the program, BASIC adds spaces making the line too long. Split the line into several smaller lines.

?LINE TOO LONG TO TRANSLATE (?TLT)

Lines are translated as they are entered; the line just entered exceeds the area reserved for translating. The line is ignored. If this message is produced while BASIC is reading a program from a file, BASIC stops reading the file. Split the line into several smaller lines.

?MISSING SUBPROGRAM (?MSP)

The CALL statement specifies a nonexistent routine name. Ensure that the name is typed correctly (it must consist of upper case letters).

?NEGATIVE SQUARE ROOT (?NGS)

Nonfatal. The expression in the SQR (square root) function has a negative value. The function returns a value of 0. BASIC continues execution of the program.

?NESTED FOR STATEMENTS WITH SAME CONTROL VARIABLE (?FSV)

A FOR statement specifies the same control variable as that specified by a FOR NEXT loop that the FOR statement is inside. Change one of the control variables to a different variable name (in both the FOR and the corresponding NEXT statement).

?NEXT WITHOUT FOR (?NWF)

A NEXT statement is without a corresponding FOR statement. Ensure that each loop starts with a FOR statement and ends with a NEXT statement which specifies the same variable. This error message is also produced if control is transferred into the middle of a loop. FOR NEXT loops should only be entered by executing the FOR statement.

?NOT ENOUGH ROOM (?NER)

There is not enough room for the FILESIZE specified. Delete unused files with the UNSAVE command.

?NUMBERS AND STRINGS (?NSM)

String and numeric values appear in the same expression or they are set equal to each other; for example, A$=2. Change either the data type of the variable (e.g., A=2) or the expression (e.g., A$="2") so that they are consistent.

?OUT OF DATA (?OOD)

The data list is exhausted and a READ statement requests additional data or the end of a file is reached and the INPUT # statement requests additional data. Ensure that there is sufficient data or test for the end-of-file condition with the IF END statement.

?PRINT USING ERROR (?PRU)

There is an error in the PRINT USING statement caused when the format specification is not a valid string, or is null, or does not contain one valid field. The error is also caused when an attempt is made to print a numeric value in a string field, a string value in a numeric field, or a negative number in a floating asterisk or floating dollar sign field that does not also specify a trailing minus sign. The message is also printed if the items in the list are not separated by commas or semicolons.

?PROGRAM TOO BIG (?PTB)

The line just entered causes the program to exceed the user area in memory; the line is ignored. Reduce program size. If this error occurs when BASIC is reading a program from a file, BASIC stops reading the file.

?RESEQUENCE ERROR (?RES)

Resequencing the program would cause lines to overlap or existing lines to be deleted, or would create an illegal line number. Reenter the command with different arguments.

?RETURN WITHOUT GOSUB (?RWG)

A RETURN is encountered before execution of a GOSUB statement. Do not transfer control to a subroutine except by executing a GOSUB or an ON GOSUB statement.

?STRING STORAGE OVERFLOW (?SSO)

Not enough memory is available to store all the strings used in the program. Reduce program size.

?STRING TOO LONG (?STL)

The maximum length of a string in a BASIC statement is 255 characters. Split string into several smaller strings.

?SUBSCRIPT OUT OF BOUNDS (?SOB)

The subscript computed is less than zero or is outside the bounds defined in the DIM statement. Ensure that expression specifying the subscript is in the correct range.

?SUBSTITUTE ERROR (?SUB)

There was no separator between the strings in the SUB command or the command would create an immediate mode statement. Retype SUB command.

?SYNTAX ERROR (?SYN)

BASIC has encountered an unrecognizable element. Common examples of syntax errors are misspelled commands, unmatched parentheses, and other typographical errors. This message can also be produced by attempting to read in a program from a file containing illegal characters, in which case BASIC stops reading the file. Retype program line or ensure that file contains a valid BASIC program.

?TOO MANY GOSUBS (?TMG)

More than 20 GOSUBS have been executed without a corresponding RETURN statement. Change the program logic so that less GOSUB statements are executed.

?TOO MANY ITEMS IN COMMON (?TIC)

There are more than 255 variable and array names in COMMON (A, A(100), A%, A%(10, 10), A$, and A$(5) are all considered different names). Reduce the number of items in COMMON by converting individual variables to elements of an array or by passing fewer items to the next program segment.

?UNDEFINED FUNCTIONS (?UFN)

A user-defined function has been used and not defined. Define the function. A function is defined only after the RUN command or CHAIN statement is executed.

?UNDEFINED LINE NUMBER (?ULN)

The line number specified in an IF, GO TO, GOSUB, ON GO TO, ON GOSUB, or CHAIN statement does not exist anywhere in the program. Ensure that the line number specified exists in the program.

?UNDIMENSIONED ARRAY IN CALL (?UAC)

The first reference to an undimensioned array appears in a CALL statement. Dimension the array with the DIM statement.

?USE REPLACE

Saving the program would have caused an existing file to be deleted. Use either a different file specification or the REPLACE command.

?VIRTUAL ARRAY CHANNEL ALREADY IN USE (?VCU)

The DIM # statement specifies a channel number which has already appeared in a DIM # statement. Specify another channel number.

Function Error Messages

Using BASIC functions improperly causes error messages to be printed. The following list names the functions and describes under which conditions BASIC functions produce errors.

All functions

The argument used is the wrong type. For example, the argument is numeric and the function expects a string expression. This condition produces ?ARGUMENT ERROR (?ARG).

All functions

The wrong number of arguments is used in a function, or the wrong character is used to separate them. For example, PRINT SIN (X,Y) produces a syntax error because the SIN function has only one argument. This condition produces ?SYNTAX ERROR (?SYN).

ASC(string)

String is not a 1-character string. This condition produces ?ARGUMENT ERROR (?ARG).

BIN(string)

Character other than blank, 0, or 1 in string or value is greater than 2^16. This condition produces ?ARGUMENT ERROR (?ARG).

CHR$(expr)

Expression is not in the range 0 to 32767. This condition produces ?ARGUMENT ERROR (?ARG).

EXP(expr)

Value of expression is greater than 87. This condition produces ?EXPONENTIATION ERROR (?EER).

FNletter

The function FNletter is not defined (function cannot be defined by an immediate mode statement). This condition produces ?UNDEFINED FUNCTION (?UFN).

LOG(expr)

Expression is negative or 0. The function returns a value of 0. This condition produces ?BAD LOG (?BLG).

LOG10(expr)

Expression is negative or 0. The function returns a value of 0. This condition produces ?BAD LOG (?BLG).

OCT(string)

Character other than blank or digits 0 through 7 appears in string, or value is greater than 2^16. These conditions produce ARGUMENT ERROR (?ARG).

PI

An argument is included. This condition produces ?SYNTAX ERROR (?SYN).

SEG$(string,expr1,expr2)

No additional error conditions.

SQR(expr)

Expression is negative. The function returns a value of 0. This condition produces ?NEGATIVE SQUARE ROOT (?NGS).

TAB(expr)

Expression is not in the range 0 to 32767. This condition produces ?ARGUMENT ERROR (?ARG).

VAL(string)

String is not a numeric constant. This condition produces ?ARGUMENT ERROR (?ARG).

APPENDIX B

FORTRAN/RT-11 LANGUAGE SUMMARY

B.1 EXPRESSION OPERATORS

Operators in each type are shown in order of descending precedence.

Type	Operator		Operates Upon
Arithmetic	**	exponentiation	arithmetic or logical constants, variables, and expressions
	*,/	multiplication, division	
	+,-	addition, subtraction unary plus and minus	
Relational	.GT.	greater than	arithmetic or logical constants, variables, and expressions (all relational operators have equal priority)
	.GE.	greater than or equal to	
	.LT.	less than	
	.LE.	less than or equal to	
	.EQ.	equal to	
	.NE.	not equal to	
Logical	.NOT.	.NOT.A is true if and only if A is false	logical or integer constants, variables, and expressions
	.AND.	A.AND.B is true if and only if A and B are both true	
	.OR.	A.OR.B is true if and only if either A or B or both are true	
	.EQV.	A.EQV.B is true if and only if A and B are both true or A and B are both false	(precedence same as .XOR.)
	.XOR.	A.XOR.B is true if and only if A is true and B is false or B is true and A is false	(precedence same as .EQV.)

B.2 STATEMENTS

The following summary of statements available in the PDP-11 FORTRAN language defines the general format for the statement. If more detailed information is needed, refer to the PDP-11 FORTRAN Language Reference Manual (DEC-11-LFLRA-C-DN1).

Statement Formats	Effect
ACCEPT	See READ, Formatted Sequential See READ, List-Directed

Arithmetic/Logical Assignment

v=e

v — is a variable name or an array element name.

e — is an expression.

The value of the arithmetic or logical expression is assigned to the variable.

Arithmetic Statement Function

f([p[,p]...])=e

f — is a symbolic name.

p — is a symbolic name.

e — is an expression.

Creates a user-defined function having the variables p as dummy arguments. When referenced, the expression is evaluated using the actual arguments in the function call.

ASSIGN s TO v

s — is an executable statement label.

v — is an integer variable name.

Associate the statement number s with the integer variable v for later use in an assigned GO TO statement.

BACKSPACE u

u — is an integer variable or constant.

The currently open file on logical unit u is backspaced one record.

BLOCK DATA [nam]

nam is a symbolic name.

Specifies the subprogram which follows as a BLOCK DATA subprogram.

CALL s[([a][,[a]]...)]

s is a subprogram name.

a is an expression, a procedure name, or an array name.

Calls the SUBROUTINE subprogram with the name specified by s, passing the actual arguments a to replace the dummy arguments in the SUBROUTINE definition.

CLOSE (p[,p]...)

p is one of the following forms:

```
UNIT =e
DISPOSE = 'SAVE'     or  DISP = 'SAVE'
DISPOSE = 'KEEP'     or  DISP = 'KEEP'
DISPOSE = 'DELETE'   or  DISP = 'DELETE'
DISPOSE = 'PRINT'    or  DISP = 'PRINT'
ERR = s
```

e is a numeric expression.
s is an executable statement label.

Closes the specified file.

COMMON [/[cb]/] nlist [[,]/[cb]/nlist]...

cb is a common block name.

nlist is a list of one or more variable names, array names, or array declarators separated by commas.

Reserves one or more blocks of storage space under the name specified to contain the variables associated with that block name.

CONTINUE

Causes no processing.

DATA nlist/clist/[[,] nlist/clist/]...

nlist is a list of one or more variable names, array names, or array element names separated by commas. Subscript expressions must be constant.

clist is a list of one or more constants separated by commas, each optionally preceded by j*, where j is a nonzero, unsigned integer constant.

Causes elements in the list of values to be initially stored in the corresponding elements of the list of variable names.

DECODE (c,f,b[,ERR=s])[list]

c is an integer expression.

f is a FORMAT statement label or array name.

b is a variable name, array name, or array element name.

s is an executable statement label.

list is an I/O list.

Changes the elements in the I/O list from character into internal format; c specifies the number of characters, f specifies the format, and b is the name of the entity containing the characters to be converted.

DEFINE FILE u(m,n,U,v)[,u(m,n,U,v)]...

u is an integer variable name or integer constant.

m is an integer variable name or integer constant.

n is an integer variable name or integer constant.

v is an integer variable name.

Defines the record structure of a direct access file where u is the logical unit number, m is the number of fixed length records in the file, n is the length in words of a single record, U is a fixed argument, and v is the associated variable.

DIMENSION a(d)[,a(d)]...

a(d) is an array declarator.

Specifies storage space requirements for arrays.

DO s [,] i = e1,e2[,e3]

s is the label of an executable statement.

i is a variable name.

ei are integer expressions.

To execute the DO loop:

1. Set i = e1
2. Execute statements through statement number s
3. Evaluate i = i+e3

4. Repeat 2 through 3 for

MAX (1, INT((e2 - e1)/e3) + 1)

iterations

ENCODE (c,f,b[,ERR=s])[list]

c is an integer expression.

f is a FORMAT statement label or an array name.

b is a variable name, array name, or array element name.

s is an executable statement label.

list is an I/O list.

Changes the elements in the list of variables into characters; c specifies the number of characters in the buffer, f specifies the format statement number, and b is the name of the entity to be used as a buffer.

END

Delimits a program unit.

END FILE u

u is an integer variable or constant.

An end-file record is written on logical unit u.

END=s,ERR=s

s is an executable statement label.

(Transfer of Control) on end-of-file or error condition is an optional element in each type of I/O statement allowing the program to transfer to statement number s on an end-of-file (END=) or error (ERR=) condition.

EQUIVALENCE (nlist)[,(nlist)]...

nlist is a list of two or more variable names, array names, or array element names separated by commas. Subscript expressions must be constant.

Each of the names (nlist) within a set of parentheses is assigned the same storage location.

EXTERNAL v[,v]...

v is a subprogram name.

Defines the names specified as FUNCTION or SUBROUTINE subprograms.

FIND (u'r)

u is an integer variable name or integer constant.

r is an integer expression.

Positions the file on logical unit u to record r and sets associated variable to record number r.

FORMAT (field specification,...)

Describes the format in which one or more records are to be transmitted; a statement label must be present.

[typ] FUNCTION nam[*n][([p[,p]...])]

typ is a data type specifier.

nam is a symbolic name.

*n is a data type length specifier.

p is a symbolic name.

Begins a FUNCTION subprogram, indicating the program name and any dummy argument names, p. An optional type specification can be included.

GO TO s

s is an executable statement label.

(Unconditional GO TO) Transfers control to statement number s.

GO TO (slist)[,] e

slist is a list of one or more executable statement labels separated by commas.

e is an integer expression.

(Computed GO TO) Transfers control to the statement label specified by the value of expression e. (If e=1 control transfers to the first statement label. If e=2 it transfers to the second statement label. etc.) If e is less than 1 or greater than the number of statement labels present, no transfer takes place.

GO TO v [[,](slist)]

v is an integer variable name.

slist is a list of one or more executable statement labels separated by commas.

(Assigned GO TO) Transfers control to the statement most recently associated with v by an ASSIGN statement.

```
IF (e) s1,s2,s3
```

e — is an expression.

si — are executable statement labels.

(Arithmetic IF) Transfers control to statement number si depending upon the value of the expression. If the value of the expression is less than zero, transfer to s1; if the value of the expression is equal to zero, transfer to s2; if the value of the expression is greater than zero, transfer to s3.

```
IF (e) st
```

e — is an expression.

st — is any executable statement except a DO or logical IF statement.

(Logical IF) Executes the statement if the logical expression is true.

```
IMPLICIT typ (a[,a]...)[,typ(a[,a]...)]...
```

typ — is a data type specifier.

a — is either a single letter, or two letters in alphabetical order separated by a dash (i.e., x-y).

The elements a represent single (or a range of) letter(s) whose presence as the initial letter of a variable specifies the variable to be of that type.

```
OPEN (p[,p]...)
```

p — is one of the following forms:

```
UNIT = e
NAME = n
TYPE = 'OLD'
TYPE = 'NEW'
TYPE = 'SCRATCH'
TYPE = 'UNKNOWN'
ACCESS = 'SEQUENTIAL'
ACCESS = 'DIRECT'
ACCESS = 'APPEND'
READONLY
FORM = 'FORMATTED'
FORM = 'UNFORMATTED'
RECORDSIZE = e
ERR = s
BUFFERCOUNT = e
INITIALSIZE = e
EXTENDSIZE = e
NOSPANBLOCKS
SHARED
DISPOSE = 'SAVE'   or DISP = 'SAVE'
DISPOSE = 'KEEP'   or DISP = 'KEEP'
DISPOSE = 'DELETE' or DISP = 'DELETE'
DISPOSE = 'PRINT'  or DISP = 'PRINT'
```

```
ASSOCIATEVARIABLE = v
CARRIAGECONTROL = 'FORTRAN'
CARRIAGECONTROL = 'LIST'
CARRIAGECONTROL = 'NONE'
MAXREC = e
BLOCKSIZE = e
```

e is an integer expression.
s is an executable statement label.
v is an integer variable name.
n is an array name, variable name, array element name, or alphanumeric literal.

Opens a file on the specified logical unit according to the parameters specified by the keywords.

PAUSE [disp]

disp is a decimal digit string containing one to five digits, an alphanumeric literal, or an octal constant.

Suspends program execution and prints the display, if one is specified.

PRINT See WRITE, Formatted Sequential
See WRITE, Listed-Directed

PROGRAM nam

nam is a symbolic name.

Specifies a name for the main program.

READ (u,f[,END=s][,ERR=s])[list]

READ f[,list]

ACCEPT f[,list]

u is an integer variable or constant.

f is a FORMAT statement label or an array name.

s is an executable statement label.

list is an I/O list.

(Formatted Sequential) Reads one or more logical records from unit u and assigns values to the elements in the list, converted according to format specification f.

READ(u[,END=s][,ERR=s])[list]

u is an integer variable or constant.

s is an executable statement label.

list is an I/O list.

(Unformatted Sequential) Reads one unformatted record from unit u, and assigns values to the elements in the list.

```
READ(u'r[,ERR=s])[list]
```

u is an integer variable or constant.

r is an integer expression.

s is an executable statement label.

list is an I/O list.

(Unformatted Direct Access) Reads record r from unit u, and assigns values to the elements in the list.

```
READ (u,*[,END=s][,ERR=s])list
```

```
READ *,list
```

```
ACCEPT *,list
```

u is an integer variable or constant.

* denotes list-directed formatting.

s is an executable statement label.

list is an I/O list.

(List-Directed) Reads one or more logical records from unit u and assigns values to the elements in the list, converted according to the data type of the list element.

```
RETURN
```

Returns control to the calling program from the current subprogram.

```
REWIND u
```

u is an integer variable or constant.

Repositions logical unit u to the beginning of the currently opened file.

```
STOP [disp]
```

disp is a decimal digit string containing one to five digits, an alphanumeric literal, or an octal constant.

Terminate program execution and print the display, if one is specified.

SUBROUTINE nam[([p[,p]...])]

nam is a symbolic name.

p is a symbolic name.

Begins a SUBROUTINE subprogram, indicating the program name and any dummy argument names, p.

TYPE

See WRITE, Formatted Sequential
See WRITE, List-Directed

Type Declaration

typ v[,v]...

typ is a data type specifier.

v is a variable name, array name, function or function entry name, or an array declarator. The name can optionally be followed by a data type length specifier (*n).

The symbolic names, v, are assigned the specified data type in the program unit.

typ is one of:

DOUBLE PRECISION
COMPLEX
COMPLEX*8
REAL
REAL*4
REAL*8
INTEGER
INTEGER*2
INTEGER*4
BYTE
LOGICAL
LOGICAL*1
LOGICAL*4

VIRTUAL a(d) [,a(d)]...

a(d) is an array declarator that specifies storage space for a VIRTUAL array.

WRITE (u,f[,ERR=s])[list]

PRINT f[,list]

TYPE f[,list]

u is an integer variable or constant.

f is a FORMAT statement label or an array name.

s is an executable statement label.

list is an I/O list.

(Formatted Sequential) Writes one or more logical records to unit u containing the values of the elements in the list, converted according to format specification f.

```
WRITE (u[,ERR=s])[list]
```

u is an integer variable or constant.

s is an executable statement label.

list is an I/O list.

(Unformatted Sequential) Writes one unformatted record to unit u containing the values of the elements in the list.

```
WRITE (u'r[,ERR=s]) [list]
```

u is an integer variable or constant.

r is an integer expression.

s is an executable statement label.

list is an I/O list.

(Unformatted Direct Access) Writes record r to unit u containing the values of the elements in the list.

```
WRITE(u,*[,ERR=s])list

PRINT *,list

TYPE *,list
```

u is an integer variable or constant.

* denotes list-directed formatting.

s is an executable statement label.

list is an I/O list.

(List-Directed) Writes one or more logical records to unit u containing the values of the elements in the list, converted according to the data type of the list element.

B.3 FORTRAN LIBRARY FUNCTIONS

FORM	ARGUMENT TYPE	RESULT TYPE	DEFINITION
ABS(X)	Real	Real	Real absolute value
IABS(I)	Integer	Integer	Integer absolute value
DABS(X)	Double	Double	Double precision absolute value

FORM	ARGUMENT TYPE	RESULT TYPE	DEFINITION
CABS(Z)	Complex	Real	Complex to Real, absolute value where z=(x,y) CABS(Z)=$(x^2+y^2)^{1/2}$
FLOAT(I)	Integer	Real	Integer to Real conversion
IFIX(X)	Real	Integer	Real to Integer conversion IFIX(X) is equivalent to INT(X)
SNGL(X)	Double	Real	Double to Real conversion
DBLE(X)	Real	Double	Real to Double conversion
REAL(Z)	Complex	Real	Complex to Real conversion, obtain real part
AIMAG(Z)	Complex	Real	Complex to Real conversion, obtain imaginary part
CMPLX(X,Y)	Real	Complex	Real to Complex conversion CMPLX(X,Y)=X+i*Y
	Truncation functions return the sign of the argument * largest integer ≤ |arg|		
AINT(X)	Real	Real	Real to Real truncation
INT(X)	Real	Integer	Real to Integer truncation
IDINT(X)	Double	Integer	Double to Integer truncation
	Remainder functions return the remainder when the first argument is divided by the second.		
AMOD(X,Y)	Real	Real	Real remainder
MOD(I,J)	Integer	Integer	Integer remainder
DMOD(X,Y)	Double	Double	Double precision remainder
	Maximum value functions return the largest value from among the argument list; > 2 arguments.		
AMAX0(I,J...)	Integer	Real	Real maximum from Integer list
AMAX1(X,Y,...)	Real	Real	Real maximum from Real list
MAX0(I,J,...)	Integer	Integer	Integer maximum from Integer list
MAX1(X,Y,...)	Real	Integer	Integer maximum from Real list
DMAX1(X,Y,...)	Double	Double	Double maximum from Double list
	Minimum value functions return the smallest value from among the argument list; > 2 arguments.		
AMIN0(I,J,...)	Integer	Real	Real minimum of Integer list
AMIN1(X,Y,...)	Real	Real	Real minimum of Real list
MIN0(I,J,...)	Integer	Integer	Integer minimum of Integer list
MIN1(X,Y,...)	Real	Integer	Integer minimum of Real list
DMIN1(X,Y...)	Double	Double	Double minimum of Double list

FORM	ARGUMENT TYPE	RESULT TYPE	DEFINITION
	The transfer of sign functions return (sign of the second argument) * (absolute value of first argument).		
SIGN(X,Y)	Real	Real	Real transfer of sign
ISIGN(I,J)	Integer	Integer	Integer transfer of sign
DSIGN(X,Y)	Double	Double	Double precision transfer of sign
	Positive difference functions return the first argument minus the minimum of the two arguments.		
DIM(X,Y)	Real	Real	Real positive difference
IDIM(I,J)	Integer	Integer	Integer positive difference
	Exponential functions return the value of e raised to the argument power.		
EXP(X)	Real	Real	e^x
DEXP(X)	Double	Double	e^x
CEXP(Z)	Complex	Complex	e^z
ALOG(X)	Real	Real	Returns log(e)(X)
ALOG10(X)	Real	Real	Returns log10(X)
DLOG(X)	Double	Double	Returns log(e)(X)
DLOG10(X)	Double	Double	Returns log10(X)
CLOG(Z)	Complex	Complex	Returns log(e) of complex argument
SQRT(X)	Real	Real	Square root of Real argument
DSQRT(X)	Double	Double	Square root of Double precision argument
CSQRT(Z)	Complex	Complex	Square root of Complex argument
SIN(X)	Real	Real	Real sine
DSIN(X)	Double	Double	Double precision sine
CSIN(Z)	Complex	Complex	Complex sine
COS(X)	Real	Real	Real cosine
DCOS(X)	Double	Double	Double precision cosine
CCOS(Z)	Complex	Complex	Complex cosine
TANH(X)	Real	Real	Hyperbolic tangent
ATAN(X)	Real	Real	Real arc tangent
DATAN(X)	Double	Double	Double precision arc tangent
ATAN2(X,Y)	Real	Real	Real arc tangent of (X/Y)
DATAN2(X,Y)	Double	Double	Double precision arc tangent of (X/Y)
CONJG(Z)	Complex	Complex	Complex conjugate, if Z=X+i*Y CONJG(Z)=X-i*Y

FORM	ARGUMENT TYPE	RESULT TYPE	DEFINITION
RAN(I,J)	Integer	Real	Returns a random number of uniform distribution over the range 0 to 1. I and J must be integer variables and should be set initially to 0. Resetting I and J to 0 regenerates the random number sequence. Alternate starting values for I and J will generate different random number sequences.

APPENDIX C

CAMERA ORIENTATION

Two entries in the Patient Study Plan (see Chapter 4) call for switch settings on the scintillation camera. These entries refer to the orientation and rotation switches on cameras manufactured by Searle Radiographics, Inc. (formerly Nuclear-Chicago).

On the Searle cameras, the 4-position Orientation switch and the 2-position Rotation switch combine to produce the eight possible coordinate relations according to the following table, in which X and Y represent the camera's field coordinates, and X' and Y' represent the coordinates of the matrix displayed on the screen.

Orientation Switch Pos.	Rotation Switch Position	
	1 (HORIZONTAL)	2 (UPRIGHT)
1	X'=-Y Y'=X	X'=X Y'=-Y
2	X'=-Y Y'=-X	X'=X Y'=Y
3	X'=Y Y'=-X	X'=-X Y'=Y
4	X'=Y Y'=X	X'=-X Y'=-Y

The camera's coordinates are oriented as shown below, with the viewer standing above the camera, and the camera rotated 180 so that it is face upwards toward the viewer.

C.1 TRANSFORMATION OPERATORS

The transformations tabulated above can be written in terms of matrix operators, thus

$$Z' = O \; R \; Z$$

where Z' = (X',Y') is the display coordinate operator

Z = (X,Y) is the camera coordinate operator

(R) [m=1 or 2] is the Rotation Switch Setting

(O) [n=1,2,3, or 4] is the Orientation Switch Setting

The matrix operators for each value of m and n are as follows:

$$(O_1) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad (O_2) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (O_3) = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \quad (O_4) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$(R_1) = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \quad (R_2) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Note that the operator (O) is cyclically permutable, so that an absolute definition of the camera's coordinate system is not necessary.

Example:

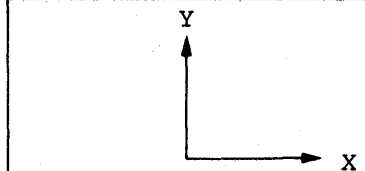
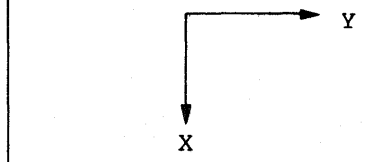
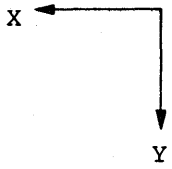
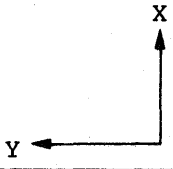
For n=4, m=1

$$Z' = \begin{bmatrix} X' \\ Y' \end{bmatrix} = O \quad R\ Z = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

$$= \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} -Y \\ -X \end{bmatrix} \begin{bmatrix} Y \\ X \end{bmatrix}$$

C.2 CONVERSION TO OTHER CAMERAS

The two switches of the Searle cameras produce any of the eight possible coordinate orientations. Corresponding functions of other camera types can be matched to the settings of these two switches. For example, given a camera that can only rotate the image clockwise, the corresponding switch settings would be as follows:

Camera X	Searle Switch Settings	
	Rotation Switch	Orientation Switch
	2 Upright	2
	1 Horizontal	3
	2 Upright	4
	1	1 Horizontal

APPENDIX D

USING A NEW DISK

Before using a new RK05 disk cartridge, you must:

1. Format the cartridge.
2. Initialize the directory (required for RT-11).

Before using a new RK06, RK07, or RL01 disk, you need only initialize the directory.

D.1 FORMATTING A NEW RK05 DISK ON AN 11/34

The following instructions detail the procedure for formatting an RK05 disk.

1. Mount the disk to be formatted in Unit 0. The following formatting procedure will work only on Unit 0.
2. Begin the boot procedure. Press CNTRL/HALT, then CNTRL/BOOT. The four numbers will appear on the console followed by a dollar sign ($).
3. At the $, type:

   L SP 1000(RET)

   where (SP) represents a space and (RET) represents RETURN. The L stands for Load address.
4. At the next $ prompt type:

   D(SP) 12737(RET)
   D(SP) 6003(RET)
   D(SP) 177404(RET)
   D(SP) 105737(RET)
   D(SP) 177404(RET)
   D(SP) 100375(RET)
   D(SP) 137(RET)
   D(SP) 1000(RET)

   Each line will be preceded by the $ prompt. The D stands for Deposit (at the address).
5. To check that you have entered the correct numbers, type:

   L(SP) 1000(RET)
   E

After you type E (SP), the system will type the input number that is deposited at location 1000.

E (SP) 001000 012737

If 012737 does not appear for location 1000, correct the location by typing the following at the $ prompt:

D (SP) 12737(RET)

Proceed to check each location and number in sequence by typing the following for each number you entered in step 4.

E (SP)

The locations and numbers should appear.

6. After you have verified that you typed in the numbers correctly, type:

L (SP) 1000 (RET)
S(RET)

Wait 60 seconds while the disk is formatted. When the disk light stops flashing, the disk is formatted and ready for use.

D.2 INITIALIZING AN RK05 RT-11 DISK DIRECTORY

Initializing a disk sets up and completely clears its file directory. A new (unused) disk must always be initialized before it is first used. The effect of an INITIALIZE operation is to remove all filenames from the directory. Thus, before you initialize any disk, be sure that there are no files on it that you might later want.

After formatting an RK05 disk, reload the GAMMA-11 system disk in Unit 0, write protected and load the RK05 disk to be initialized in Unit 1. Type:

CRTL/C
INIT (SP) RK1:(RET)

The system will respond with:

RK1:/Init are you sure?

The system prompt RK1:/Init are you sure? is always printed to provide an opportunity for you to verify the command. Typing a Y followed by RETURN initiates the operation, while N followed by RETURN ignores the operation and returns control to the monitor command mode. Check your command line, make sure you are initializing the correct disk, and then type a Y followed by RETURN.

D.3 INITIALIZING AN RK06 RT-11 DISK DIRECTORY

As with RK05 disks, initializing an RK06 disk sets up and completely clears its file directory. A new (unused) disk must always be initialized before it is first used. The effect of an INITIALIZE operation is to remove all filenames from the directory. Thus, before you initialize any disk, be sure that there are no files on it that you might later want.

To initialize an RK06 disk, load the GAMMA-11 system disk, write protected and type:

```
CTRL/C
INIT/BAD (SP) DM1:(RET)
```

The system will respond with:

```
DM1:/Init are you sure?
```

The system prompt DM1:/Init are you sure? is always printed to provide an opportunity for you to verify the command. Typing a Y followed by RETURN initiates the operation, while N followed by RETURN ignores the operation and returns control to the monitor command mode. Check your command line, make sure you are initializing the correct disk, and then type a Y followed by RETURN.

The /BAD option on the INIT command makes sure that any bad blocks on the disk are designated in the directory as FILE.BAD. In this way, the bad blocks are removed from the available disk blocks, thus minimizing disk errors.

INDEX

Administrative data block, 2-2 to 2-4, 2-7
Assembling GAMMA-11, 4-1

Backing up disks, 1-8, 1-9
BACKUP, 1-8, 1-9
BASIC commands,
  APPEND, A-8
  CLEAR, A-8
  COMPILE, A-8
  DEL, A-8
  LENGTH, A-8
  LIST, A-8
  NEW, A-8
  OLD, A-8
  RENAME, A-8
  REPLACE, A-8
  RESEQ, A-8
  RUN, A-9
  SAVE, A-9
  SCR, A-9
  SUB, A-9
  UNSAVE, A-9
BASIC error messages, 3-11 to 3-14, A-10 to A-18
BASIC functions,
  ABORT, A-7
  ABS, A-5
  ASC, A-6
  ATN, A-5
  BIN, A-6
  CHR$, A-6
  CLK$, A-6
  COS, A-5
  CTRLC, A-7
  CTRLO, A-7
  DAT$, A-6
  EXP, A-5
  INT, A-5
  LEN, A-6
  LOG, A-5
  LOG10, A-5
  OCT, A-6
  PI, A-5
  POS, A-6
  RCTRLC, A-7
  RND, A-5
  SEG$, A-7
  SGN, A-5
  SIN, A-5
  SQR, A-6
  STR$, A-7
BASIC functions (Cont.)
  SYS, A-7
  TAB, A-6
  TRM$, A-7
  TTYSET, A-7
  VAL, A-7
BASIC statements,
  CALL, A-1
  CHAIN, A-1
  CLOSE, A-1
  COMMON, A-1
  DATA, A-1
  DEF, A-2
  DIM, A-2
  END, A-2
  FOR, A-2
  GO TO, A-2
  GOSUB, A-2
  IF, A-2
  IF TO, A-2
  INPUT, A-3
  KILL, A-3
  LET, A-3
  LINPUT, A-3
  NAME, A-3
  NEXT, A-3
  ON, A-3
  OPEN, A-3
  OVERLAY, A-4
  PRINT, A-4
  RANDOMIZE, A-4
  READ, A-4
  REM, A-4
  RESET, A-4
  RESTORE, A-4
  RETURN, A-4
  STOP, A-4
BASIC support routines, 3-1
BGAMMA routines, 3-21
Building GAMMA-11, 1-1

Camera switch settings, C-1
Comment block, 2-2 to 2-4, 2-6
COMPRS, 2-7
Configuring disks, 1-4
CTRL/C, A-9
CTRL/O, A-9
CTRL/Q, A-9
CTRL/S, A-9
CTRL/U, A-10
Curve,
  dynamic, 2-17

DELETE, A-10
Disk formatting, D-1
Disks,
  backing up, 1-8, 1-9
  configuring, 1-4
  initializing, D-2, D-3
  RK05, 1-1, 1-2, D-1, D-2
  RK06, 1-1, 1-2, D-3
  RK07, 1-1, 1-2
  RL01, 1-1, 1-2
Distribution media, 1-1, 1-2
DL, 1-2
DM, 1-2
Dynamic curve, 2-17
Dynamic study, 2-2, 2-17

Error messages,
  BASIC, 3-11 to 3-14, A-10 to A-18
  FORTRAN, 3-11 to 3-14

FADOFF, 2-7
FGADM1 routines, 3-20
FGADM2 routines, 3-21
FGCDM1 routines, 3-21
FGCOM2 routines, 3-21
FGPICK routines, 3-7, 3-21
FGPLOT routines, 3-21
FGPTOV routines, 3-21
File,
  dynamic patient, 2-2
  patient, 2-1, 2-10, 3-2
File names,
  patient, 2-10
File type, 2-1
Files,
  internal, 2-17
  macro, 2-18
  playback, 2-18
Formatting,
  disk, D-1
FORTRAN arithmetic operators, B-1
FORTRAN error messages, 3-11 to 3-14
FORTRAN library functions,
  ABS, B-11
  AIMAG, B-12
  AINT, B-12
  ALOG, B-13
  ALOGID, B-13
  AMAX0, B-12
FORTRAN library functions (Cont.)
  AMAX1, B-12
  AMIN0, B-12
  AMIN1, B-12
  AMOD, B-12
  ATAN, B-13
  ATAN2, B-13
  CABS, B-12
  CCOS, B-13
  CEXP, B-13
  CLOG, B-13
  CMPLX, B-12
  CONJG, B-13
  COS, B-13
  CSIN, B-13
  CSQRT, B-13
  DABS, B-11
  DATAN, B-13
  DATAN2, B-13
  DBLE, B-12
  DCOS, B-13
  DEXP, B-13
  DIM, B-13
  DLOG, B-13
  DLOG10, B-13
  DMAX1, B-12
  DMIN1, B-12
  DMOD, B-12
  DSIGN, B-13
  DSIN, B-13
  DSQRT, B-13
  EXP, B-13
  FLOAT, B-12
  FORM, B-12, B-13
  IABS, B-11
  IDIM, B-13
  IDINT, B-12
  IFIX, B-12
  INT, B-12
  ISIGN, B-13
  MAX0, B-12
  MAX1, B-12
  MIN0, B-12
  MIN1, B-12
  MOD, B-12
  RAN, B-14
  REAL, B-12
  SIGN, B-13
  SIN, B-13
  SNGL, B-12
  SQRT, B-13
  TANH, B-13
FORTRAN logical operators, B-1
FORTRAN relational operators, B-1

FORTRAN statements,
  ACCEPT, B-2, B-8, B-9
  ASSIGN, B-2
  assignment, B-2
  BACKSPACE, B-2
  BLOCK DATA, B-3
  CALL, B-3
  CLOSE, B-3
  COMMON, B-3
  CONTINUE, B-3
  DATA, B-3
  DECODE, B-4
  DEFINE FILE, B-4
  DIMENSION, B-4
  DO, B-4
  ENCODE, B-5
  END, B-5
  END FILE, B-5
  END=, B-5
  EQUIVALENCE, B-5
  ERR=, B-5
  EXTERNAL, B-5
  FIND, B-6
  FORMAT, B-6
  FUNCTION, B-6
  GO TO, B-6
  IF, B-7
  IMPLICIT, B-7
  OPEN, B-7
  PAUSE, B-8
  PRINT, B-8, B-10, B-11
  PROGRAM, B-8
  READ, B-8, B-9
  RETURN, B-9
  REWIND, B-9
  STOP, B-9
  SUBROUTINE, B-10
  TYPE, B-10, B-11
  VIRTUAL, B-10
  WRITE, B-10, B-11
FORTRAN support routines,
    3-1, 3-10
  supplemental, 3-19
Functions,
  IBYTE, 3-20
  ISPR, 3-20
  LBYTE, 3-20
  RDPI, 3-20
  RDPR, 3-20
  RSPI, 3-20

GAM routines, 3-10
GASP routines, 3-7
GCHR routines, 3-10
GCVG routines, 3-6
GCVP routines, 3-6
GDIS routines, 3-6
GMXG routines, 3-6
GMXP routines, 3-6
GPAR routines, 3-3
GPAW routines, 3-3
GPDR routines, 3-3
GPDW routines, 3-3
GPF routines, 3-3
GPFR routines, 3-3
GPFW routines, 3-3
GPKX routines, 3-7
GPKY routines, 3-7
GPLR routines, 3-3
GPLW routines, 3-4
GPMR routines, 3-3
GPMW routines, 3-3
GPOV routines, 3-7
GSAG routines, 3-7
GSAR routines, 3-6
GSAW routines, 3-7
GSVG routines, 3-6
GSVP routines, 3-6

IBYTE functions, 3-20
IFGFRM routines, 3-21
IGLSTR routines, 3-21
IGLSTW routines, 3-21
Indirect command files, 4-1,
    4-5
Initializing disks, D-2,
    D-3
Internal files, 2-17
ISPR functions, 3-20

LBYTE functions, 3-20
Linking FORTRAN routines,
    3-11
Linking GAMMA-11, 4-1, 4-5
List mode study, 2-6

Macro files, 2-18
Magnetic tape, 1-1, 1-3
Matrix data, 2-2, 2-15
MDOFF, 2-7
Messages,
  BASIC error, 3-14
  FORTRAN error, 3-14
Multiple static study, 2-4

NADOFF, 2-7

PADOFF, 2-7
Patient file, 2-1, 2-10, 3-2
  dynamic, 2-2
Patient file names, 2-10
Playback files, 2-18

RDPI functions, 3-20
RDPR functions, 3-20
Region of interest, 2-15, 2-16
RK, 1-2
RK05 disks, 1-1, 1-2, D-1, D-2
RK06 disks, 1-1, 1-2, D-3
RK07 disks, 1-1, 1-2
RL01 disks, 1-1, 1-2
Routines,
  BASIC support, 3-1
  BGAMMA, 3-21
  FGADM1, 3-20
  FGADM2, 3-21
  FGCDM1, 3-21
  FGCOM2, 3-21
  FGPICK, 3-7, 3-21
  FGPLOT, 3-21
  FGPTOV, 3-21
  FORTRAN support, 3-1, 3-10
  GAM, 3-10
  GASP, 3-7
  GCHR, 3-10
  GCVG, 3-6
  GCVP, 3-6
  GDIS, 3-6
  GMXG, 3-6
  GMXP, 3-6
  GPAR, 3-3
  GPAW, 3-3
  GPDR, 3-3
  GPDW, 3-3
Routines (Cont.)
  GPF, 3-3
  GPFR, 3-3
  GPFW, 3-3
  GPKX, 3-7
  GPKY, 3-7
  GPLR, 3-3
  GPLW, 3-4
  GPMR, 3-3
  GPMW, 3-3
  GPOV, 3-7
  GSAG, 3-7
  GSAR, 3-6
  GSAW, 3-7
  GSVG, 3-6
  GSVP, 3-6
  IFGFRM, 3-21
  IGLSTR, 3-21
  IGLSTW, 3-21
  supplemental FORTRAN support, 3-19
RSPI functions, 3-20

Save area, 2-11, 2-15, 2-17, 3-6
Save area descriptor block, 2-11, 3-6
Single static study, 2-3
Study,
  dynamic, 2-2, 2-17
  list mode, 2-6
  multiple static, 2-4
  single static, 2-3
Supplemental FORTRAN support routines, 3-19
Switch settings,
  camera, C-1
SYSGEN, 1-1, 1-4

Z-count area, 2-3, 2-2, 2-6
ZCTOFF, 2-7

GAMMA-11
System Reference
AA-2186B-TC

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify)_____

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code or Country_____

Please cut along this line.

---------- Fold Here ----------

---------- Do Not Tear - Fold Here and Staple ----------

FIRST CLASS
PERMIT NO. 152
MARLBOROUGH, MA
01752

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

Software Documentation
200 Forest Street MR1-2/E37
Marlborough, Massachusetts 01752

digital

EJ 05726 77 05A/S20 23 075

Printed in U.S.A.