MU BASIC/RT-11 SYSTEM INSTALLATION GUIDE

DEC-11-LIBMA-A-D

MU BASIC/RT-11
SYSTEM INSTALLATION GUIDE

DEC-11-LIBMA-A-D

MU BASIC/RT-11 VØ1-Ø1

Order additional copies as directed on the Software Information page at the back of this document.

digital equipment corporation · maynard. massachusetts

First Printing, May 1975

The HOW TO OBTAIN SOFTWARE INFORMATION page, located at the back of this document, explains the various services available to DIGITAL software users.

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.

CONTENTS

TABLES

PREFACE

This document contains the information necessary to build and maintain an MU BASIC/RT-11 system. It serves both as an introduction to MU BASIC/RT-11 and as a reference manual for maintaining the system. Any user who is building, running, linking, or assembling MU BASIC/RT-11 or who is writing assembly language routines to be linked with MU BASIC/RT-11 should read this manual.

Chapter 1 of this manual contains the information to get started with MU BASIC/RT-11. The procedures described in this chapter are usually sufficient to build a working MU BASIC/RT-11 system, but the inclusion of certain options (described in section 1.3.1) requires procedures described in other chapters. The documentation conventions used in this manual are described in section 1.3.2.

In this manual the term BASIC[1] can refer to either the BASIC language or the MU BASIC/RT-11 system and the term system manager means any user who is installing the system.

There are several other documents associated with this one. They are:

Document	Order Number
BASIC-11 Language Reference Manual	DEC-11-LIBBA-B*-D
MU BASIC/RT-11 User's Manual	DEC-11-LIBRA-A*-D
Getting Started with RT-11	DEC-11-ORCPA-D*-D
RT-11 System Reference Manual	DEC-11-ORUGA-B*-D
Software Performance Summary	DEC-11-XSPSA-A*-D

The BASIC-11 Language Reference Manual describes the features of the language, and the MU BASIC/RT-11 User's Manual describes the procedures

---

[1]BASIC is a registered trademark of the Trustees of Dartmouth College.
*This column represents the revision and is subject to change.

to use a working MU BASIC/RT-11 system and some enhancements added to the language for this system. An MU BASIC/RT-11 user should have access only to these two documents unless the user is:

1. running, linking, or assembling MU BASIC/RT-11 with the RT-11 Monitor,
2. writing assembly language routines to be linked with MU BASIC/RT-11,
3. using RT-11 programs other than MU BASIC/RT-11, or
4. altering the log-on files.

CHAPTER 1

GETTING STARTED WITH MU BASIC/RT-11

1.1 INTRODUCTION TO MU BASIC/RT-11

MU BASIC/RT-11 is a multi-user BASIC-11[1] system operating under the RT-11 Monitor. MU BASIC/RT-11 can be run under either the single-job (SJ) version or Foreground/Background (F/B) version of the RT-11 Monitor. Under the F/B Monitor MU BASIC/RT-11 may be run either as the foreground job or as the background job.

Up to eight users can run programs simultaneously. Each user has a fixed area in memory in which BASIC programs can be created. The BASIC system executes a section of each user's program in a round-robin manner.

An optional log-on feature, HELLO, restricts system access to authorized users. An optional file protection system allows a user to create files that are protected from other BASIC users. Because the optional file protection system is internal to MU BASIC/RT-11 there is no file protection from any other RT-11 program. For example, if MU BASIC/RT-11 is running in the Foreground and PIP (Peripheral Interchange Program) is running in the background, any program file or data file can be copied by PIP or printed on the terminal or line printer.

Each time MU BASIC/RT-11 is run, a once-only initial dialogue occurs during which the system is configured. The configuration process allows tradeoffs of software features and peripheral access against system performance. An optional configuration file feature allows a user to save the initial dialogue to avoid having to complete the dialogue each subsequent time BASIC is run.

[1]BASIC-11 is DIGITAL's name for a family of BASIC systems for the PDP-11 computers.

1.1.1 Overview of the Software Kit

The binary MU BASIC/RT-11 Software Kit is available on four media: DECpack, DECtape, cassette, and paper tape. Each kit contains user documentation and the materials necessary to build a complete MU BASIC/RT-11 system. The components of an MU BASIC/RT-11 Software Kit are inventoried on checklists attached to the outside of the kit. It is recommended that the contents of the package be verified against the checklist; any discrepancies should be reported to DIGITAL's Software Distribution Center (SDC) in Maynard (address given in section 1.2).

The software provided on DECpack, DECtape, and cassette contains "running" versions of MU BASIC/RT-11. After the files provided have been transferred to the system device, a running version of MU BASIC/RT-11 can be loaded and executed by the RT-11 RUN or FRUN command. The software provided on paper tape must be linked to create a running version. Section 1.3 contains the instructions to start an MU BASIC/RT-11 system from any of the binary MU BASIC/RT-11 kits.

The contents of the MU BASIC/RT-11 kit can be divided into five logical groups.

1. Running versions of MU BASIC/RT-11 (not in paper tape kits)
2. BASIC programs and data files
3. Object files
4. Source files
5. Special RT-11 Monitor files

The files provided are in standard RT-11 format and have a filename followed by a period and an extension. RT-11 directories are attached to each cassette, DECtape, and DECpack provided. A filename is on each paper tape provided. Each logical group can be identified by the extension of the files.

The running versions have the extension "SAV" or "REL". The following versions are provided.

File Descriptor	Features Contained
MUBAS.SAV	All optional features, overlaid system, linked for SJ or Background.
MUBAS.REL	All optional features, overlaid system, linked for Foreground.

Files with an extension "BØØ" are special BASIC files that allow users to start and terminate their session with the system. The file, "lUSER.DØØ" is a special data file that is used to demonstrate the initial dialogue. The files provided are:

File Descriptor	Function
INIT.BØØ	Initialization program that does not include the log-on feature.
INITH.BØØ	Initialization program that includes the optional log-on feature.
BYE.BØØ	BYE command program that does not include the log-on feature.
BYEH.BØØ	BYE command program that includes the optional log-on feature.
HELLO.BØØ	Program that contains the code for the log-on procedure.
PASWRD.BØØ	File containing the "user id's" and passwords used in the log-on procedure.
NOTICE.BØØ	File containing the message to be typed when a user logs-on to the system.
lUSER.DØØ	Special configuration file used during the system demonstration.
EXIT.BØØ	Program to exit to the RT-11 Monitor.
ZAP.BØØ	Program to delete obsolete password files.

MU BASIC/RT-11 may be linked from the object files. This is done to exclude some optional features or to include user-written assembly language routines or special arithmetic hardware support. It is necessary to link the object files provided on paper tape to create a running version. The object files have an extension of ".OBJ" except for the special arithmetic hardware support modules. The object files provided are:

File	Function
MUBA.OBJ	RT-11 location interface module.
MUBI.OBJ	Assembly language routine interface module.
MUBT.OBJ	Timer module.
MUBS1.OBJ MUBS2.OBJ	Multi-user and RT-11 I/O modules.
MUBS2E.OBJ	Multi-user and RT-11 I/O module that is linked after BASIC Edit module.
MUBVSJ.OBJ	Standard terminal vector address module.
MUBVFB.OBJ	Special Foreground/Background vector address module excluding the console terminal.
MUBM1.OBJ	Arithmetic module - no special hardware.
MUBM2.OBJ	Transcendental functions (SIN, COS, LOG, etc.) module - no special hardware.
MUBM1.EAE	Arithmetic module - EAE hardware.
MUBM2.EAE	Transcendental functions module - EAE hardware.
MUBM1.EIS	Arithmetic module - EIS hardware.
MUBM2.EIS	Transcendental functions module - EIS hardware.
MUBM1.FIS	Arithmetic module - FIS hardware.
MUBM2.FIS	Transcendental functions module - FIS hardware.
MUBM1.FPU	Arithmetic module - FPU hardware.
MUBM2.FPU	Transcendental functions module - FPU hardware.
MUBNM2.OBJ	Arithmetic module that excludes support for transcendental functions.
MUBH.OBJ	Optional-function module with string features.
MUBHNS.OBJ	Optional-function module without string features.
MUBR.OBJ	Root section of BASIC language interpreter with string features.
MUBRNS.OBJ	Root section of BASIC language interpreter without string features.
MUBE.OBJ	BASIC Edit module with string features.
MUBENS.OBJ	BASIC Edit module without string features.

File	Function
MUBET.OBJ	BASIC Edit Terminator module.
MUBX.OBJ	BASIC execute module with string features.
MUBXNS.OBJ	BASIC execute module without string features.
MUBXT.OBJ	BASIC execute Terminator module.
MUBPR.OBJ	PRINT USING statement processor (for Root section) with string features.
MUBPX.OBJ	PRINT USING statement processor (for execute section) with string features.
MUBPXN.OBJ	PRINT USING statement processor (for execute section) without string features.
MUBNP.OBJ	Module with no PRINT USING processor.
MUBC.OBJ	CALL statement (for assembly language subroutines) processor module.
MUBNC.OBJ	Module with no CALL statement processor.
MUBO.OBJ	Initialization modules.
MUBZ.OBJ	Last module linked with BASIC.

To link assembly language routines it is necessary to edit and assemble the provided source files before linking BASIC. The source files provided have an extension ".MAC".

File	Function
MUBI.MAC	Assembly language routine and background routine interface module.
GETARG.MAC	Optional routine to assist assembly language routines in receiving arguments and passing results.

The MU BASIC/RT-11 system requires a special RT-11 cassette handler which is provided on DECpack, DECtape, and cassette as file "CT.NEW" and on paper tape as "CT.OBJ". Instructions to change cassette handlers are provided in section 1.3.3.

In addition to the binary software kits described above, MU BASIC/RT-11 Source Kits and Listing Kits are available to assist in system development and modification. See Chapter 5 for more information on the source kit.

1.1.2 Supported Hardware

MU BASIC/RT-11 requires a minimum hardware system for RT-11, and, in addition, a minimum of 16K of memory and a KW11-L real-time clock (or equivalent). To run MU BASIC/RT-11 in a Foreground/Background environment requires 28K of memory.

MU BASIC/RT-11 supports all peripheral devices supported by the RT-11 VØ2 Monitor except the VT-11 display screen. In addition MU BASIC/RT-11 also supports up to eight terminal interfaces, which can include any mixture of the following:

DL11A	
DL11B	
DL11C	
DL11D	
DL11E	used in conjunction with a modem for dial-up lines

All hardware-supported speeds are acceptable.

The terminals supported are: LA30, LA36, VT05, VT50, LT33 (also known as ASR33), and LT35 (ASR35). There are some restrictions on lower-case support and special terminal instructions support - all lower-case input is converted to upper case and all alphanumeric display cursor control characters must be programmed by the user.

Although all the peripheral devices supported by RT-11 are supported by MU BASIC/RT-11, in certain system configurations memory size may restrict the use of some RT-11 peripheral devices.

NOTE

MU BASIC/RT-11 does not support the LPS-11 Laboratory Peripheral System, the VT-11 display processor (GT-40 or GT-44), or the LV-11 Electrostatic Printer/Plotter in the plotter mode.

DECtape based RT-11 systems are not recommended for MU BASIC/RT-11.

MU BASIC/RT-11 should not be run with PDP-11 systems that do not include a real-time

clock. Systems without clocks are subject to the following restrictions: individual terminals may be subject to static (the affected terminal will be incapable of input or output) and remote terminals may not be able to disconnect from the system. Either of these conditions can be corrected only by exiting to the monitor and reloading the system.

1.2 SERVICES

This section describes some of the software services available from DIGITAL. These include training, SPR (Software Performance Report) system, Digital Software News, software and document distribution, DECUS (Digital Equipment Corporation Users Society), Software Consulting Services, and software registration. Some additional information is included in the HOW TO OBTAIN SOFTWARE INFORMATION pages at the back of this manual.

Training

A variety of hardware and software courses are offered by DIGITAL's Educational Services Group as detailed in the Educational Courses Catalog (available from the Software Distribution Center). These courses are excellent vehicles for learning about both general PDP-11 programming and the use of PDP-11 software. "Hands on" training using PDP-11 family systems is a particularly valuable feature of most courses and seminars.

SPR System

The SPR (Software Performance Report) system is the mechanism by which MU BASIC/RT-11 users can report software problems and suggestions for improvements. SPRs are acknowledged when received in Maynard, and an individual answer is returned to the sender as soon as possible. If the SPR reports a software problem, the answer will include a patch or alternate procedure, if possible.

NOTE

Documentation errors and inadequacies should be reported on the READER'S COMMENTS page at the end of this manual, not on an SPR.

Before sending an SPR to DIGITAL, the user should make certain that the problem is reproducible and that a correction for the problem has not already been published in the Digital Software News or Software Performance Summary. If the problem is new, fill out a Software Performance Report and send it to:

Software Communications
Post Office Box F
Maynard, Massachusetts 01754

The SPR should include as much documentation as possible to help describe and isolate the problem. It must include configuration information, software version numbers, and any examples, tapes and listings that may be necessary for us to investigate a problem or suggested change. In general the response time is shortened by additional information provided.

In addition to software problems, SPRs are useful for reporting suggestions and comments on MU BASIC/RT-11. SPRs are monitored by DIGITAL management, and are considered by the development groups when MU BASIC/RT-11 changes are made.

Blank SPR forms are included in software kits, and additional forms are available from the Software Distribution Center. Replacement forms are included with each answer.

Digital Software News for the PDP-11

Announcements of new and revised software as well as programming notes, software problems and solutions, and documentation corrections are published monthly in the Digital Software News. Filling out and mailing to DIGITAL the MU BASIC/RT-11 registration form included in the MU BASIC/RT-11 kit assures the user of receiving this publication for one year.

Software and Document Distribution

The PDP-11 Software Price List contains a complete list of programs and documents currently available. Items may be ordered directly from the Software Distribution Center by using the Software Order Form enclosed in the Price List. As noted previously, new and revised software is announced via the Digital Software News.

DECUS

Digital Equipment Corporation Users Society (DECUS) was established to advance the effective use of Digital Equipment Corporation's computers and peripheral equipment. It is a voluntary, non-profit users group supported by DIGITAL. DECUS's objectives are to:

1. Advance the art of computation through mutual education and interchange of ideas and information
2. Establish standards and provide channels to facilitate the free exchange of computer programs among members
3. Provide feedback to the manufacturer on equipment and programming needs.

The Society sponsors technical symposia twice a year (Spring and Fall) in the U.S., and once a year in Europe, Canada, and Australia. It maintains a Program Library, publishes a library catalog, proceedings of symposia, and a periodic newsletter: DECUSCOPE.

A DECUS-Europe organization was formed in 1970 to assist in the servicing of European members.

Users interested in joining DECUS must obtain and complete a registration form. Forms can be obtained from the nearest DIGITAL sales office or by writing the appropriate Administrative office.

The Administrative office is located at Digital Equipment Corporation, Maynard, Massachusetts 01754, and all correspondence should be directed to the attention of the DECUS Executive Director.

The European Regional Administrative office is located at:

DECUS EUROPE
Digital Equipment Corporation International (Europe)
P. O. Box 340
1211 Geneva 26
Switzerland

Software Consulting Services

DIGITAL maintains a staff of programmers and consultants whose services are available to DIGITAL customers for a fee. Through DIGITAL's Software Consulting Services, customers have been able to reduce development costs and still obtain quality customized software. Areas of

expertise include process control, data communications, data analysis, information retrieval, numerical control, direct digital control, typesetting, simulation, commercial data processing and special purpose timesharing.

Registration

By completing and returning the registration form included in the MU BASIC/RT-11 kit, the user is eligible to order new updates of this software at the prevailing MU BASIC/RT-11 Update Kit prices plus any handling or shipping charges. The user must register to be eligible. It is important that the MU BASIC/RT-11 registration form be mailed as well as the RT-11 registration form. Complete and mail the forms to:

Digital Equipment Corporation
Software Distribution Center
Building 1-2
146 Main Street
Maynard, Massachusetts 01754

1.3 MU BASIC/RT-11 SYSTEM BUILD

1.3.1 Options in the System Build Procedure

BASIC has many optional and variable features that allow the software to be optimized for the particular needs of the installation. To build a system as quickly and simply as possible, follow the procedures in this chapter. Alternatively the BASIC software can be optimized by procedures described in other chapters such as varying the responses to the initial dialogue, linking BASIC from the object modules, including assembly language routines with BASIC, and reassembling BASIC from the source kit. This section describes the options and associated tradeoffs in all of these.

The procedures in this chapter produce a version of BASIC that includes string features and the PRINT USING statement and that allows access only to the RT-11 system device. The string features included are string variables, functions, and operations and string virtual array files. PRINT USING is a special formatting PRINT statement. Only three decisions must be made (for the procedures in this chapter).

One is the possible inclusion of the HELLO feature. The HELLO feature restricts access of the system to authorized users and enables the file protection system. The HELLO feature has no effect on the amount of memory available for user programs or the speed of program execution.

The second choice is the mode in which BASIC is run. BASIC can be run in any of three ways:

1. under the SJ Monitor
2. as a background job under the F/B Monitor
3. as a foreground job under the F/B Monitor

More memory is available to user programs when BASIC is run under the SJ Monitor because the monitor is smaller and there is no other job. Under the F/B Monitor BASIC performs well in the background when the foreground job is frequently I/O-bound; however, an execute-bound foreground job will lock out (not allow execution of) BASIC. To run MU BASIC/RT-11 and perform RT-11 program development simultaneously, it is necessary to run BASIC in the foreground. In the foreground BASIC may slow the execution of the background job but does not lock out the background even when one or more BASIC users are execute-bound.

The third decision is the maximum number of BASIC users on the system. Additional users reduce the amount of memory and the CPU time allocated to each user. Under certain circumstances the overall efficiency of the system may be reduced by permitting additional users.

There are several options in the initial dialogue that are not described in this chapter. They are: access to devices other than the system device, use of terminals at nonstandard I/O addresses, and exclusion of optional functions. See Chapter 2 for a description of the options available and the tradeoffs involved.

The BASIC load modules provided are in an overlaid form (some sections of code that are not needed simultaneously share the same area in memory) but it is possible to link BASIC (Chapter 4) in a memory-resident, nonoverlaid form. The overlaid form provides more memory for user programs but in certain circumstances decreases the system's performance. If the system includes approximately 3000 words of memory that are not needed for user programs, the performance of the system (with more than one user) can generally be improved by linking a nonoverlaid BASIC.

BASIC must be linked from the object modules to build a version without string features or the PRINT USING statement processor. Versions of BASIC without string features do not have any string variables, operations, or functions and allow only string constants in PRINT, CALL, and file control statements. Excluding the PRINT USING statement processor increases the available memory in a nonoverlaid version of BASIC by approximately 500 words but does not increase available memory for overlaid versions. Excluding the string features increases the available memory in a nonoverlaid version by approximately 900 words and in an overlaid version by approximately 275 words. Excluding the string features also frees 350 words of memory (in all versions) occupied by the string optional functions, but these can be excluded even if string features are included (see Chapter 2).

BASIC must also be linked to take advantage of any special hardware arithmetic processors or to eliminate support for transcendental functions. The hardware arithmetic processors allow much faster evaluation of arithmetic expressions and provide more room for user programs but do not increase the precision of the calculations. Eliminating the transcendental functions makes the following functions unavailable: SIN, COS, LOG, LOG10, EXP, and ATN functions (and the expression A↑B where B is not an integer or is greater than 256). More memory (approximately 570 words with no arithmetic hardware) is available to user programs when the transcendental functions are excluded.

To include user-written assembly language routines (Chapter 3), it is necessary to link BASIC. These routines can provide features otherwise unavailable to the BASIC users.

BASIC can be assembled from the source kit. The options available are selectively excluding transcendental functions and specifying the longer form of the error messages.

1.3.2 Documentation Conventions

The documentation conventions used in this manual are different from the conventions in the BASIC-11 Language Reference Manual because this manual contains specific step-by-step instructions while the conventions of the BASIC-11 Language Reference Manual describe general formats.

Conventions, abbreviations and standards used include:

(1) All numbers are listed in decimal unless otherwise indicated.

(2) The following abbreviations are used:

CTRL	Control key
CR	Return key
LF	Line feed key
integer	Any positive integral number in the range described in the text.

(3) <CR> or <LF> indicate that the RETURN or LINE FEED key should be typed at that place in the dialogue.

(4) <ALT> indicates that the ALTMODE (or ESCAPE) key should be typed at that place in the dialogue.

(5) <CTRL/X> indicates that the CONTROL key should be pressed and held down while another key, "X" is also pressed.

(6) <TAB> indicates that a horizontal tab should be typed.

(7) On ASR33 and ASR35 Teletype[1] terminals, special characters that are produced by holding down one key and depressing another are:

↑	SHIFT/N
\	SHIFT/L
[	SHIFT/K
]	SHIFT/M
<TAB>	CTRL/I

(8) The sample terminal dialogue provided in this document contains version numbers where they would normally appear. The version numbers given include "xx's" in those fields that may vary from installation to installation. The exact contents of these fields are not of interest (except when reporting software problems on SPRs), as long as appropriate digits appear in the area indicated in this document. The same is true for "FREE CORE" messages printed by any of the system programs and for "FREE BLOCKS" messages included in device directories.

---

[1]Teletype is a registered trademark of the Teletype Corporation.

NOTE

No RT-11 system program ever HALTs with the expectation that the CONTINUE switch can be pressed to resume operation after corrective action has been taken. If the computer HALTs (the RUN light is off), a significant error has occurred and the entire section should be repeated from the beginning.

In case of error messages not explained in this document, please refer to the RT-11 SYSTEM REFERENCE MANUAL and the MU BASIC/RT-11 USER'S MANUAL.

If user errors occur within a section, go back to the beginning of that particular section.

Typing errors may be corrected using the standard RT-11 input editing techniques (RUBOUT and CTRL/U).

The responses listed in the demonstration do not occur immediately but are printed when the appropriate operation is completed.

1.3.3 System Build

This section contains the step-by-step instructions to build a working MU BASIC/RT-11 system and the information needed to maintain the system.

Users who are building an MU BASIC/RT-11 system from the software kit should start at Step 1.

Users who already have a working MU BASIC/RT-11 system should read:

Step 9 for instructions to run BASIC

Step 8 for instructions to change the log-on files

Step 15 for instructions to return control to the RT-11 monitor

Step 11 for restrictions on foreground/background operation

Before any action is taken with the software kit the following should be read completely: Chapters 1 and 2 of this manual, the BASIC-11 Language Reference Manual, the MU BASIC/RT-11 User's Manual, and the Software Performance Summary. Special attention should be given to Section 1.4 of this manual and to any information pertinent to RT-11 or

MU BASIC/RT-11 in the Software Performance Summary -- any recommended changes to the software or the documents should be made immediately to avoid difficulties.

The MU BASIC/RT-11 system operates under the control of the RT-11 Monitor. It is necessary to have a working RT-11 system (version 2 or later) before attempting any of the procedures described in this document. Getting Started With RT-11 describes the procedures to build a working RT-11 system from the RT-11 software kits. The RT-11 System Reference Manual should also be available for reference. It is need for certain operations, such as initialization of devices, and for understanding any error messages produced by the RT-11 CUSPs (Common User System Programs -- LINK, EDIT, MACRO, PIP, etc.).

STEP 1

If the RT-11 Monitor has already been bootstrapped go to Step 2.

NOTE

If the console terminal is a serial LA30 DECwriter or is a VT05 operating at greater than 300 baud the RT-11 console handler should be altered as described in the RT-11 System Reference Manual.

After the system has been built (see the GETTING STARTED WITH RT-11 document), the monitor can be loaded into memory from disk or DECtape as follows:

1. Press HALT
2. Mount the system device on unit 0. Note that if disk unit 0 is disabled, RT-11 can be loaded from another RK disk unit using the software bootstrap described later in this section.
3. If a disk is being used, be sure the WRITE PROTECT light is lit.
4. If a DECtape unit is being used, set the WRITE ENABLE/WRITE LOCK switch to WRITE LOCK and the REMOTE/OFF/LOCAL switch to REMOTE.

If the hardware configuration includes the BM792-YB hardware bootstrap:

1. Set the Switch Register to 173100 (the address of the ROM Bootstrap Loader).
2. Press the LOAD ADDR switch.
3. Set the Switch Register to the address of the word count register of disk or DECtape on which the monitor resides:

177462	for	RF11 disk
177406	for	RK11,RK05 disk
177344	for	DECtape

4. Press the START switch.

If the hardware configuration includes the MR11-DB hardware bootstrap:

1. Set the Switch Register to:

773100	for	RF11 disk
773110	for	RK11, RK05 disk
773120	for	DECtape

2. Press the LOAD ADDR switch.
3. Press the START switch.

If neither hardware bootstrap is available, or if an RK disk unit other than 0 is to be used as the system device, one of the following bootstraps must be entered manually using the switch register. First set the switch register to 1000 and press the LOAD ADDR switch. Then set the switch register to the first value shown for the appropriate bootstrap and raise the DEPosit switch. Continue depositing the values shown.

DECtape	Disk (RK11,RK05)	RK Disk (other than Unit 0)	Disk (RF11)
12700	12700	12700	12700
177344	177406	177406	177466
12710	12710	12760	5010
177400	177400	xxxxxx *	5040
12740	12740	4	12740
4002	5	12700	177400
5710	105710	177406	12740
100376	100376	12710	5
12710	5007	177400	105710
3		12740	100376
105710		5	5007
100376		105710	
12710		100376	
5		5007	
105710			
100376			
5007			

```
* xxxxxx =  20000 for unit 1
            40000 for unit 2
            60000 for unit 3
           100000 for unit 4
           120000 for unit 5
           140000 for unit 6
           160000 for unit 7
```

When all the values have been entered, set the switches to 1000 and press the LOAD ADDR and START switches.

The monitor loads into memory and prints one of the following identification messages followed by a dot (.) on the terminal:

```
RT-11SJ V02-XX
RT-11FB V02-XX
```

The message printed indicates which monitor (Single-Job or F/B) has been loaded; the user may determine which is to be loaded during the system build operation. See Step 9 for directions on how to bring up the alternate monitor while under control of the one currently running.

After the message has printed, the system device should be WRITE ENABLEd. The monitor is ready to accept keyboard commands.

STEP 1

Write enable the system device.

STEP 2

Enter the current date with the RT-11 DAT command. The date format is dd-mmm-yy where dd is the day of the month, mmm are the first three characters of the month, and yy is the year (for example, 12-FEB-75).

```
Type:       DAT dd-mmm-yy<CR>
Response:   .
```

Run RT-11 PIP (Peripheral Interchange Program).

```
Type:       R PIP<CR>
Response:   *
```

If MU BASIC/RT-11 is provided on DECtape go to Step 3.

If MU BASIC/RT-11 is provided on cassette go to Step 4.

If MU BASIC/RT-11 is provided on paper tape go to Step 5.

To transfer the MU BASIC/RT-11 files from the master MU BASIC/RT-11 disk to the RT-11 system disk:

Mount the MU BASIC/RT-11 system disk (DEC-11-LMUBA-A-HC) on Unit 1 WRITE PROTECTED. (Instructions for mounting a disk can be found in Getting Started With RT-11.)

Scan the master disk for readability on this drive:

Type: RK1:/K<CR>

Each block on the disk is read and checked for errors; the process takes about four minutes.

Response: *

If the response is anything but the above, the drive needs alignment in order to read the master disk.

Transfer the files to the RT-11 system disk.

Type: *.*=RK1:*.*/X<CR>

Response: *

Go to Step 6.

STEP 3

The following instructions describe transferring the MU BASIC/RT-11 files from the master DECtape (DEC-11-LMUBA-A-UC) to the RT-11 System device.

Mount the MU BASIC/RT-11 System DECtape (DEC-11-LMUBA-A-UC) on Unit 0, WRITE PROTECTED. (For information on how to mount a DECtape, see Getting Started With RT-11.)

Type: *.*=DT:*.*/X<CR>

Response: *

Dismount the DECtape.

Go to Step 6.

STEP 4

Prior to inserting any master cassette during this procedure, Write Protect the data by placing the orange tabs on the bottom of the cassette such that the holes are uncovered. Cassette unit 0 is on the left and cassette unit 1 is on the right.

To transfer the MU BASIC/RT-11 files from the master cassettes (DEC-11-LMUBA-A-TC1 to DEC-11-LMUBA-A-TC5) to the RT-11 system device.

Insert the MU BASIC/RT-11 System Cassette Tape 1 of 5 (DEC-11-LMUBA-A-TC1) into unit 0.

```
Type:       *.*=CT0:*.*/X/M:1<CR>
Response:   *
```

Rewind and dismount the cassette and mount the MU BASIC/RT-11 System Cassette Tape 2 of 5 (DEC-11-LMUBA-A-TC2) into unit 0.

```
Type:       *.*=CT0:*.*/X/M:1<CR>
Response:   *
```

Rewind and dismount the cassette. If BASIC is not to be relinked from the object modules go to Step 6.

If BASIC is to be linked from the object modules, mount the MU BASIC/RT-11 System Cassette Tape 3 of 5 (DEC-11-LMUBA-A-TC3) into unit 0.

```
Type:       *.*=CT0:*.*/X/M:1<CR>
Response:   *
```

Rewind and dismount the cassette and mount the MU BASIC/RT-11 System Cassette Tape 4 of 5 (DEC-11-LMUBA-A-TC4) into unit 0.

```
Type:       *.*=CT0:*.*/X/M:1<CR>
Response:   *
```

Rewind and dismount the cassette and mount the MU BASIC/RT-11 System Cassette Tape 5 of 5 (DEC-11-LMUBA-A-TC5) into unit 0.

```
Type:       *.*=CT0:*.*/X/M:1<CR>
Response:   *
```

Rewind and dismount the cassette and store all the master cassettes in a safe place.

Go to Step 6.

STEP 5

To transfer the MU BASIC/RT-11 files from the master paper tapes (DEC-11-LMUBA-A-PR1 to DEC-11-LMUBA-A-PR38) and DEC-11-LMUBA-A-PA1 to DEC-11-LMUBA-A-PA12) to the RT-11 system device.

Place the paper tape labeled "MUBA.OBJ" (DEC-11-LMUBA-A-PR1) in the reader. Press the FEED button until blank leader is over the read head.

For this, and all the following paper tape instructions, the convention xxxxxx.xxx is used to represent the name contained on the tape label underneath the MU BASIC/RT-11 Paper Tape number. In the first example, the command below will appear as "MUBA.OBJ=PR:/B<CR>".

```
Type:       xxxxxx.xxx=PR:/B<CR>
Response:   *
```

During these operations, a ?CHKSUM? message printed during any paper tape operation means an input error has occurred. Retry the tape if such a message is printed.

Remove the tape from the paper tape reader take up bin.

Repeat this operation for the following paper tapes:

Name	Number
MUBI.OBJ	DEC-11-LMUBA-A-PR2
MUBT.OBJ	DEC-11-LMUBA-A-PR3
MUBS1.OBJ	DEC-11-LMUBA-A-PR4
MUBS2.OBJ	DEC-11-LMUBA-A-PR5

Name	Number
MUBS2E.OBJ	DEC-11-LMUBA-A-PR6
MUBVSJ.OBJ	DEC-11-LMUBA-A-PR7
MUBVFB.OBJ	DEC-11-LMUBA-A-PR8
MUBET.OBJ	DEC-11-LMUBA-A-PR9
MUBXT.OBJ	DEC-11-LMUBA-A-PR10
MUBNP.OBJ	DEC-11-LMUBA-A-PR11
MUBNC.OBJ	DEC-11-LMUBA-A-PR12
MUBO.OBJ	DEC-11-LMUBA-A-PR13
MUBZ.OBJ	DEC-11-LMUBA-A-PR14
MUBC.OBJ	DEC-11-LMUBA-A-PR15
MUBPR.OBJ	DEC-11-LMUBA-A-PR16

If BASIC is to be linked to include string features, repeat this operation for the following paper tapes:

Name	Number
MUBH.OBJ	DEC-11-LMUBA-A-PR17
MUBR.OBJ	DEC-11-LMUBA-A-PR18
MUBE.OBJ	DEC-11-LMUBA-A-PR19
MUBX.OBJ	DEC-11-LMUBA-A-PR20
MUBPX.OBJ	DEC-11-LMUBA-A-PR21

If BASIC is to be linked to exclude string features, repeat this operation for the following paper tapes:

Name	Number
MUBHNS.OBJ	DEC-11-LMUBA-A-PR22
MUBRNS.OBJ	DEC-11-LMUBA-A-PR23
MUBENS.OBJ	DEC-11-LMUBA-A-PR24
MUBXNS.OBJ	DEC-11-LMUBA-A-PR25
MUBPXN.OBJ	DEC-11-LMUBA-A-PR26

If BASIC is not to use any special arithmetic hardware repeat the operation for these files:

Name	Number
MUBM1.OBJ	DEC-11-LMUBA-A-PR27
MUBM2.OBJ	DEC-11-LMUBA-A-PR28

If BASIC is to use the EAE arithmetic hardware repeat the operation for these files but use the extension "OBJ" instead of "EAE":

Name	Number
MUBM1.EAE	DEC-11-LMUBA-A-PR29
MUBM2.EAE	DEC-11-LMUBA-A-PR30

If BASIC is to use the EIS arithmetic hardware repeat the operation for these files but use the extension "OBJ" instead of "EIS":

Name	Number
MUBM1.EIS	DEC-11-LMUBA-A-PR31
MUBM2.EIS	DEC-11-LMUBA-A-PR32

If BASIC is to use FIS arithmetic hardware repeat the operation for these files but use the extension "OBJ" instead of "FIS":

Name	Number
MUBM1.FIS	DEC-11-LMUBA-A-PR33
MUBM2.FIS	DEC-11-LMUBA-A-PR34

If BASIC is to use FPU arithmetic hardware repeat the operation for these files but use the extension "OBJ" instead of "FPU":

Name	Number
MUBM1.FPU	DEC-11-LMUBA-A-PR35
MUBM2.FPU	DEC-11-LMUBA-A-PR36

If BASIC is to be linked without the transcendental functions repeat this operation for the following file:

Name	Number
MUBNM2.OBJ	DEC-11-LMUBA-A-PR37

If BASIC is to use cassettes repeat this operation for the following file:

CT.OBJ	DEC-11-LMUBA-A-PR38

Place the paper tape labelled "INIT.B00" (DEC-11-LMUBA-A-PA1) in the reader. Press the FEED button until blank leader is over the read head.

For this and all the following paper tape instructions, the convention xxxxxx.xxx is used to represent the name contained on the tape label underneath the MU BASIC/RT-11 paper tape number. In the first example, the command below will appear as "INIT.B00=PR:/A<CR>".

```
Type:          xxxxxx.xxx=PR:/A<CR>
Response:      *
```

Remove the tape from the paper tape reader take up bin.

Repeat this step for the following paper tapes:

Name	Number
INITH.B00	DEC-11-LMUBA-A-PA2
BYE.B00	DEC-11-LMUBA-A-PA3
BYEH.B00	DEC-11-LMUBA-A-PA4
HELLO.B00	DEC-11-LMUBA-A-PA5
PASWRD.B00	DEC-11-LMUBA-A-PA6
NOTICE.B00	DEC-11-LMUBA-A-PA7
EXIT.B00	DEC-11-LMUBA-A-PA8
lUSER.D00	DEC-11-LMUBA-A-PA9
MUBI.MAC	DEC-11-LMUBA-A-PA10
GETARG.MAC	DEC-11-LMUBA-A-PA11
ZAP.B00	DEC-11-LMUBA-A-PA12

Link the files that have been transferred to the system device using the following instructions. The two running versions produced are the same as the running version supplied on other media (except that the math package may support special arithmetic hardware).

```
Type:          <CTRL/C>
Response:      ↑C
               .

Type:          R LINK<CR>
Response:      *

Type:          MUBAS,MUBAS=MUBA,MUBVSJ/B:400/C<CR>
Response:      *

Type:          MUBR,MUBS1,MUBS2,MUBT,MUBI/C<CR>
Response:      *

Type:          MUBNC,MUBPR/C<CR>
Response:      *

Type:          MUBM1,MUBM2/C<CR>
Response:      *
```

```
Type:       MUBE,MUBS2E,MUBET/O:1/C<CR>
Response:   *

Type:       MUBX,MUBPX,MUBXT/O:1/C<CR>
Response:   *

Type:       MUBO/O:1/C<CR>
Response:   *

Type:       MUBH,MUBZ/O:2<CR>
Response:   *

Type:       MUBAS,MUBREL=MUBVFB/R/C<CR>
Response:   *

Type:       MUBR,MUBS1,MUBS2,MUBT,MUBI/C<CR>
Response:   *

Type:       MUBNC,MUBPR/C<CR>
Response:   *

Type:       MUBM1,MUBM2/C<CR>
Response:   *

Type:       MUBE,MUBS2E,MUBET/O:1/C<CR>
Response:   *

Type:       MUBX,MUBPX,MUBXT/O:1/C<CR>
Response:   *

Type:       MUBO/O:1/C<CR>
Response:   *

Type:       MUBH,MUBZ/O:2<CR>
Response:   *
```

If the new system will use cassettes, link the new cassette handler:

```
Type:       CT.NEW=CT<CR>
Response:   *
```

In all cases,

```
Type:       <CTRL/C>
Response:   ↑C
            .

Type:       R PIP<CR>
Response:   *
```

Go to Step 6.

STEP 6

If BASIC is not to use cassettes go to Step 7. If BASIC is going to use cassettes it is necessary to replace the cassette handler supplied with RT-11 with the one provided with BASIC.

The cassette handler included with BASIC is the RT-11 V02 cassette handler with an extra SET option added. If any patches to the RT-11 cassette handler are published in Digital Software News, they should not be done to the handler supplied with BASIC unless the article specifies "the cassette handler supplied with MU BASIC/RT-11."

```
Type:       CT.OLD=CT.SYS/X/Y<CR>
Response:   *

Type:       CT.SYS/D/Y<CR>
Response:   ?REBOOT?
            *
```

The ?REBOOT? message is expected at this point and does not indicate a mistake; however, do not reboot the system at this point.

```
Type:       CT.SYS=CT.NEW/X/Y<CR>
Response:   ?REBOOT?
            *

Type:       SY:/O<CR>
Response:   RT-11xx Vxx-xx
            .

Type:       DAT dd-mmm-yy<CR>
Response:   .
```

STEP 7

To build a system without the HELLO feature go to Step 9.

BASIC is provided with the HELLO feature disabled; access to BASIC is unrestricted, and there is no file protection. The following procedure enables the HELLO feature.

```
Type:       <CTRL/C>
Response:   ↑C
            .

Type:       R MUBAS<CR>
Response:   MU BASIC/RT-11 Vxx-xx
            CONFIGURATION FILE:  OLD (O), NEW (N), OR NONE (<CR>)

Type:       O<CR>
Response:   FILE NAME --

Type:       lUSER<CR>
Response:   READY


Type:       NAME "INIT.B00" TO "INITN.B00"<CR>
Response:   READY

Type:       NAME "BYE.B00" TO "BYEN.B00"<CR>
Response:   READY

Type:       NAME "INITH.B00" TO "INIT.B00"<CR>
Response:   READY

Type:       NAME "BYEH.B00" to "BYE.B00"<CR>
Response:   READY
```

Go to Step 8.

STEP 8

This procedure updates the password file, PASWRD.B00, and the message-of-the-day file, NOTICE.B00. PASWRD.B00 and NOTICE.B00 are created

and updated as if they were BASIC program files. They can be updated by a privileged user whenever the BASIC system is operational. The files listed in this procedure are the files provided in the software kit, but any user created log-on files can be updated by the same procedure.

NOTE

PASWRD.B00 and NOTICE.B00 are subject to the same size restrictions that normal BASIC programs have. If the ?PTB (Program Too Big) error message is produced, the files must be edited on a terminal with a larger user area -- if necessary, the system can be reconfigured.

```
Type:       OLD PASWRD.B00<CR>
Response:   READY

Type:       LISTNH<CR>
Response:   10 "00,SYSMAN,SYSTEM MANAGER
            20 "AS,BASIC,COMPATIBILITY WITH BASIC/RT-11
            30 'SP,SAMPLE,NONPRIVILEGED USER
            READY
```

The existing password file allows users to get started with the user ID 00 and the password SYSMAN, the user ID AS and the password BASIC, or the user ID SP and the password SAMPLE. The first two are privileged accounts; the user ID SP is a nonprivileged account. To delete any of these three accounts simply type the line number followed by a <CR>. Do not delete the system manager account (user ID 00, password SYSMAN) as it will be used in this demonstration. To add new user ID's and passwords, type them in the general form:

```
line number ["'] user ID,password,comment
```

where:

line number	is a unique line number.
"	indicates a privileged user.
'	indicates a nonprivileged user.
user ID	can be any two alphanumeric characters (the second character can be a null or blank); must be followed by a comma.

password	can be from one to six characters; must be followed by a comma.
comment	can be any identifying comment.

NOTE

User ID's should not end with the last two letters of any RT-11 default file extension. The following pairs of characters should not be user ID's: AD, AK, MP, OR, DA, LD, ST, AC, AP, BJ, AL, EL, AV, and YS.

The HELLO feature is available only on systems that include the SYS and CHR$ functions and where each user has at least 450 words of memory.

After editing the password file, save it as the file PASWRD.N00.

```
Type:       SAVE PASWRD.N00<CR>
Response:   READY
```

Next, run the program ZAP.B00 which deletes the contents of the old password file and renames the new password file, PASWRD.N00, to PASWRD.B00 so that the system will use the new file.

```
Type:       RUN ZAP.B00<CR>
Response:   ?SOB AT LINE 20
            READY
```

The ?SOB (Subscript Out of Bounds) error message is expected and does not indicate a mistake.

```
Type:       NAME "PASWRD.N00" TO "PASWRD.B00"<CR>
Response:   READY

Type:       OLD NOTICE.B00<CR>
Response:   READY

Type:       LISTNH<CR>
Response:   10 "WELCOME TO MU BASIC/RT-11
            READY
```

This is the "message of the day" file, which is printed automatically on a user's terminal after the log-on procedure is completed.

To create a new message file,

```
Type:       SCR<CR>
Response:   READY
```

To enter a new message, type the message in the following general form.

```
Type:       line number "message<CR>
               :
            line number "message<CR>
```

where:

line number	is a unique line number. The message lines are printed in numerical order (without the line number or the double quote).
message	can contain any printing characters.

After editing the message file,

```
Type:       REPLACE NOTICE.B00<CR>
Response:   READY
```

Test the log-on procedure.

```
Type:       BYE<CR>
Response:   USERID xx LOGGED OFF  --  GOODBYE
            PLEASE SAY HELLO
```

where xx is the current user ID (AS is the default).

```
Type:       HELLO<CR>
Response:   USERID:
```

Enter a user ID for a privileged account; for example, to use the system manager's account:

```
Type:       00<CR>
Response:   PASSWORD:
```

The letters that are typed at this point do not appear on the terminal.

Type: SYSMAN<CR>

The entire contents of the message in NOTICE.B00 are printed at this point. Then BASIC prints

READY

indicating that the log-on procedure has been successful.

If the log-on files have been updated on an operational BASIC system, the procedure has been completed. Type the BYE command to terminate your session.

If the log-on files have been updated as part of the BASIC system build procedure, return control to the RT-11 Monitor.

Type: RUN EXIT.B00<CR>
Response: .

Go to Step 9.

STEP 9

There are several conditions that must be checked each time that BASIC is to be run. These concern device assignment, the cassette handler, the line printer handler, the use of the VT-11 display screen and write enabling the system device.

BASIC cannot use device name abbreviations that have been associated with a device by the RT-11 ASSIGN command. To deassign any previous assignments,

Type: ASSIGN<CR>
Response: .

BASIC cannot access multiple volume files (MVF) on cassettes. During normal RT-11 cassette operations, an output file that is too long for one cassette can be continued on another cassette. A file created in this manner is a multiple volume file. A special SET option disables the multiple volume file feature, and it must be used when BASIC accesses cassettes. To disable multiple volume files,

```
Type:       SET CT NOMVF<CR>
Response:   .
```

If this is not done, reaching the end of a cassette causes the system to enter a closed loop. One exception to this is that when BASIC is running in the foreground and is not using the console terminal multiple volume files can be used.

If the system includes a line printer, the line printer handler should be set to return an error if the line printer is offline.

```
Type:       SET LP NOHANG<CR>
Response:   .
```

This is especially necessary on systems with LS11 line printers or remote terminals.

If the system includes a VT11 display processor (GT-40 or GT-44), the RT-11 Monitor command GT ON should not be in effect. If the GT ON command has been previously entered,

```
Type:       GT OFF<CR>
Response:   .
```

An exception to this is that when BASIC is running in the foreground and is not accessing the console terminal, the VT11 display can be used. GT ON does require approximately 1.25K of memory and it is not possible to enter the GT OFF command while BASIC is running.
If the system device is write protected, write enable it.

BASIC can now be run under the SJ Monitor or as a foreground or background job under the F/B Monitor.

To run BASIC under the F/B Monitor go to Step 11.

To run BASIC under the SJ monitor, follow this procedure: If the monitor in memory is the SJ Monitor go to Step 10.

The following procedure replaces the F/B Monitor currently in memory with the SJ Monitor. For RT-11 systems where the system device is not an RK disk, substitute the appropriate monitor filenames.

```
Type:       R PIP<CR>
Response:   *
```

```
Type:       RKMNFB.SYS=MONITR.SYS/R/Y<CR>
Response:   ?REBOOT?
            *
```

The ?REBOOT? message is expected and does not indicate a mistake. Do not reboot the system at this point.

```
Type:       MONITR.SYS=RKMNSJ.SYS/R/Y<CR>
Response:   ?REBOOT?
            *
```

The ?REBOOT? message is expected and does not indicate a mistake. Do not reboot the system at this point.

```
Type:       A/U=MONITR.SYS<CR>
Response:   *
```

Reboot the system.

```
Type:       SY:/O
Response:   RT-11SJ Vxx-xx

Type:       DAT dd-mmm-yy<CR>
Response:   .
```

Go to Step 10.

STEP 10

To run BASIC under the SJ Monitor:

```
Type:       R MUBAS<CR>
Response:   MU BASIC/RT-11 V01-01
```

Go to Step 14.

STEP 11

When BASIC is run with the F/B Monitor, there are certain restrictions that must be observed.

Either BASIC or the other job can have sole access to the console terminal. They cannot share the console by means of the RT-11 CTRL/F and CTRL/B key commands. If both jobs attempt to have input or output with the console terminal, the other job is locked out by BASIC. When BASIC is run in the foreground, it usually excludes the console terminal. This allows RT-11 program development to be done in the background on the console. When BASIC is run in the background, it usually includes the console terminal and the foreground job cannot access the console.

NOTE

When BASIC is run in the foreground there is no protection from the background job. For example, if a CTRL/F and two CTRL/C's are typed at the console, BASIC is terminated. BASIC should never be terminated in this manner; consequently, it is very important that CTRL/F not be typed at the console terminal after the initial dialogue is completed.

If the foreground job uses any device (other than the system device), its handler must be loaded. Devices which do not support named files, paper tape reader/punch, line printer, and card reader should be assigned to either the foreground or the background job. If both jobs are allowed to access these devices simultaneously, the input and output of both jobs can get mixed.

If the F/B Monitor is currently in memory go to Step 12 to run BASIC as a background job or go to Step 13 to run BASIC as a foreground job.

The following procedure replaces the SJ Monitor currently in memory with the F/B Monitor. For RT-11 systems with the system device not an RK disk, substitute the appropriate monitor filenames:

```
Type:       R PIP<CR>
Response:   *

Type:       RKMNSJ.SYS=MONITR.SYS/R/Y<CR>
Response:   ?REBOOT?
            *
```

The ?REBOOT? message is expected and does not indicate an error. Do not reboot the system at this point.

```
Type:       MONITR.SYS=RKMNFB.SYS/R/Y<CR>
Response:   ?REBOOT?
            *
```

The ?REBOOT? message is expected and does not indicate an error. Do not reboot the system at this point.

```
Type:       A/U=MONITR.SYS<CR>
Response:   *
```

Reboot the system.

```
Type:       SY:/O<CR>
Response:   RT-11FB Vxx-xx

Type:       DAT dd-mmm-yy
Response:   *
```

Go to Step 12 to run BASIC as a background job.

Go to Step 13 to run BASIC as a foreground job.

STEP 12

The foreground job must be loaded before BASIC is run in the background. The foreground job can not be unloaded while BASIC is running.

```
Type:       R MUBAS<CR>
Response:   MU BASIC/RT-11 Vxx-xx
```

Go to Step 14.

STEP 13

When BASIC is run as a foreground job it should not normally use the console terminal because that is reserved for the background job. The supplied foreground load module (MUBAS.REL) has been linked with a special F/B terminal address module which excludes the console terminal.

For BASIC to access any device other than the system device, the device handler must be loaded before BASIC is run in the foreground.

A device handler can be loaded and its units assigned to the background job or the foreground job or can be loaded and available to both jobs.

For example, the RT-11 command:

```
LOAD DT, CT0=F, CT1=B, LP=F<CR>
```

makes all DECtape units available to both the background job and BASIC, cassette unit and the line printer can be accessed only by BASIC and cassette unit 1 can be accessed only by the background job. If the console terminal is available, these assignments can be dynamically changed by reissuing the LOAD command while BASIC is running. All devices must also be specified in the initial dialogue (see Chapter 2). The procedures in this chapter do not require any device handlers to be loaded.

NOTE

When BASIC is run in the foreground, the handlers for any nonsystem devices specified in the initial dialogue should be loaded and available to BASIC. If a device has been specified in the initial dialogue but the handler has not been loaded or is restricted to the other job, attempting to access this device produces the messages:

```
?DEV [AT LINE xxxxxx]
?FTS  [AT LINE xxxxxx]
READY
```

When BASIC is run in the foreground it is necessary to specify the amount of memory reserved for user program storage and I/O overhead. The maximum value that leaves enough room for background programs should be used. A recommended trial value is 8000 (for systems with 28K or more of memory). This allows 6.5K of memory for the background job (enough to run all the RT-11 CUSPs except MACRO which requires more memory).

To run BASIC in the foreground:

Type: FRUN MUBAS/N!integer<CR>

where integer is the decimal number of words of memory reserved for BASIC.

Type: <CTRL/F>
Response: F>
MU BASIC/RT-11 Vxx-xx

Go to Step 14.

STEP 14

The procedures in this section produce a BASIC system that has I/O access to the user's terminals and the system device only. If other devices are to be used or if any of the terminal I/O addresses are nonstandard or if the system device is not an RK disk or to exclude any optional functions, see Chapter 2. If any terminal is a fast serial terminal, see Chapter 2 or follow this procedure and then see the MU BASIC/RT-11 User's Manual description of the SET TTY command.

After BASIC is run, it prints:

```
MU BASIC/RT-11 Vxx-xx
CONFIGURATION FILE:  OLD (O), NEW (N), OR NONE (<CR>)?
```

Type: <CR>
Response: KEEP USR RESIDENT (Y OR N)?

Type: N<CR>
Response: ENTER AVAILABLE DEVICES:
?

Type: RK0P<CR>
Response: ?

Type: <CR>
Response: SYSTEM BUFFERS SPECIFICATION (NUMBER,SIZE(WORDS)):
?

Type: 1,256<CR>
Response: ?

Type: <CR>

Response: xxxx WORDS REMAINING
SYSTEM I/O AREA SIZE (WORDS)?

Type: 1<CR>

Response: NUMBER OF USER CHANNELS?

Type: 14<CR>

Response: NUMBER OF SYSTEM CHANNELS?

Type: 1<CR>

Response: MAXIMUM NUMBER OF CHANNELS FOR UNPRIVILEGED USER?

Type: 6<CR>

Response: MAXIMUM NUMBER OF BLOCKS IN OPEN FOR UNPRIVILEGED USER?

Type: 50<CR>

Response: DEFAULT NUMBER OF BLOCKS FOR OPEN?

Type: 25<CR>

Response: OPT FNS (N-NONE, A-ALL, OR I-INDIVIDUAL)?

Type: A<CR>

Response: xxxx WORDS CORE AVAILABLE
HOW MANY USERS?

Type: integer<CR>

where integer is the desired number of BASIC users (in the range 1 to 8)

Response: xxxx WORDS CORE AVAILABLE
xxx WORDS PER USER IF DIVIDED EVENLY
CORE #1?

Type: <CR>

Response: TERM #1?

Type: <CR>

If there are two or more users,

Response: TERM #2

Type: <CR>

Complete this procedure for each terminal.

If BASIC has been run in the foreground and is not using the console terminal,

```
Type:       <CTRL/B>
Response:   B>
```

and do not type CTRL/F at the console until after BASIC has been terminated.

BASIC should print a message and start the log-on procedure at each terminal or, if the HELLO feature has been excluded, print READY on each terminal. If this does not occur, try running BASIC and following the procedure in the initial dialogue in Chapter 2.

This completes the system build. Go to Step 15 for instructions to terminate the BASIC system.

STEP 15

To terminate BASIC, log-on under a privileged user ID (on systems without the HELLO feature all users are privileged), and

```
Type:       RUN EXIT.B00<CR>
```

or alternatively,

```
Type:       A=SYS(-3)<CR>
```

Control is returned to the RT-11 Monitor which prints the prompt character . (dot) on the console terminal.
If BASIC has used cassettes, return the cassette handler to its normal mode of operation,

```
Type:       SET CT MVF<CR>
Response:   .
```

If BASIC has been running in the foreground, unload BASIC and any handlers that have been loaded. For example,

```
Type:       UNLOAD FG,DT,CT,LP<CR>
Response:   .
```

1.4 RELEASE NOTES AND RESTRICTIONS

MU BASIC/RT-11 users should always keep abreast of MU BASIC/RT-11 related notices published by DIGITAL. Changes published in the Software Performance Summary should be made immediately. Changes published in the Digital Software News should be made as soon as possible to systems in use. In addition to continued surveillance of the above documents, the user should be aware of the following notes and restrictions at release time.

MU BASIC/RT-11 REQUIRES RT-11 V02B

MU BASIC/RT-11 requires RT-11 version 2B or later versions. For users who have RT-11 version 2 and who have not yet received version 2B BASIC can be run subject to the following restrictions:

Only the SJ Monitor can be used.

The cassette handler supplied with BASIC (CT.NEW) must not be renamed to CT.SYS as described in the system build (section 1.3.3), Step 6.

Cassettes cannot be used on-line with BASIC (they should not be included in the initial dialogue) but they can be used offline with the other RT-11 CUSPs such as PIP.

PROBLEMS WITH CASSETTE AND MAGTAPE OPERATIONS

Multiple Open Files on One Cassette Unit:

BASIC does not produce an error message when one user attempts to open more than one file on a cassette unit. However, the results are unpredictable. Consequently, any user accessing cassettes must ensure that only one file is open on a cassette unit at one time. BASIC correctly produces an error message when one user has a file open on a cassette and another user tries to open a file on that cassette.

File-structured Output to Nonfile-Structured Cassette:

File-structured output to a cassete that contains nonfile-structured data causes the ?DHE (Device Hardware Error) or ?DNE (Device Not Enabled) error message and the cassette to be zeroed. Any subsequent file-structured output can then be successfully completed.

Example:

```
10 OPEN "CTØ:" FOR OUTPUT AS FILE #1
20 PRINT #1, "NON FILE-STRUCTURED"
30 CLOSE #1
40 OPEN "CTØ:ABC" FOR OUTPUT AS FILE #1
50 PRINT #1, "FILE STRUCTURED"
60 CLOSE #1
70 PRINT "OPERATION SUCCESSFUL"
RUNNH
?DHE AT LINE 4Ø
READY
GO TO 40
OPERATION SUCCESSFUL
READY
```

This is not the recommended method of zeroing cassettes. All devices, including cassettes, should be zeroed by RT-11 PIP before using them with BASIC. See the RT-11 System Reference Manual for a description of PIP.

Certain Cassette and Magtape Operations Suspend all Users:

During certain operations on cassettes and magtapes all BASIC users are suspended. At the completion of the operation all user programs continue. Once the operation has started it cannot be terminated by the CTRL/C key command but can be terminated by taking the device offline (removing the cassette from the drive).

When a cassette file is opened for output with a nonzero mode, all users are suspended until the tape reaches the specified position. This does not happen when MODE is not specified, when opening a file for input, or when using the SAVE, REPLACE, or OLD command on cassette.

When a cassette file that has been opened for input is closed, the cassette handler reads to the end of the file. All users are suspended until the end of the file is reached. This can be a serious problem if a long cassette file is opened for input, only a small portion of the data is read, and then the file is closed.

When a magtape is being searched for a file or for the logical end-of-tape, all users are suspended until the operation is completed. The operation can take up to five minutes (for 2400 foot magtapes). This restriction of magtape operations should be considered before including magtape in the configuration.

DIRECTORY LISTING PROGRAM

Included in the MU BASIC/RT-11 binary kit is a BASIC public library program, 9CAT.B, that can be used by any BASIC user to obtain a directory listing of all files or subsets of files on the system device. The program is run by typing:

```
RUN $9CAT<CR>
```

The program prints:

```
USERID:
```

To obtain a listing of all files on the system's device, type a carriage return only. To obtain a listing of files in the public library, enter a dollar sign, followed by a carriage return. To obtain a listing of files in your group library, enter a number sign (#) followed by a carriage return. To obtain a listing of files in another group library, enter the corresponding group id (a letter or digit), followed by a carriage return. To obtain a listing of your or another user's files, enter the corresponding two character user ID, followed by a carriage return. (Actual access to the files printed by 9CAT is a function of the user's privilege status and the MU BASIC/RT-11 file protection system.) Any response other than the above including <RUBOUT> or <CTRL/U>, causes 9CAT to reprompt with the USERID message.

I/O operations by other users may, infrequently, cause the 9CAT program to produce erroneous output. If this occurs, simply type <CTRL/C> and rerun 9CAT.

A minimum user area of 730 words is required to run 9CAT.

VIRTUAL FILE SIZES AND SUBSCRIPT CHECKING

Access to all elements of an existing virtual file may not be allowed by BASIC when neither the FILESIZE nor the dimension is specified in the OPEN statement. Access in this case is allowed only to the number of blocks specified in the "DEFAULT NUMBER OF BLOCKS FOR OPEN?" question in the initial dialogue (Chapter 2, Step 5). This problem exists when the size of the virtual file is greater than the default output file size. This problem can be avoided completely by always specifying the same FILESIZE or dimension in the OPEN statement that creates the virtual file and in subsequent OPEN statements that access the existing virtual file. If a FILESIZE or dimension is specified in an OPEN FOR INPUT statement that exceeds the actual size of the virtual file, access is allowed to the entire file but the file is not extended.

Subscript checking in virtual files does not work correctly. Access is not always limited to the subscript specified in the OPEN statement. Programs should never depend on the ?SOB (Subscript Out of Bounds) error message terminating the program when the subscript specified in the OPEN statement is exceeded but should check the subscript range explicitly. Access is limited to the physical size of the file and the ?SOB error message is always produced when a virtual file element falls outside of the limits of the file.

PIP DIRECTORY LISTING SUSPENDS BASIC USR REQUESTS

When BASIC is running in the foreground and PIP is running in the background, a PIP directory listing temporarily locks out all BASIC USR requests. Any BASIC command or statement that requires the USR will not be executed until the directory listing is complete. The CTRL/C key command will not be executed while a user is suspended. The following require the USR:

OLD command
SAVE command
REPLACE command
APPEND command
BYE command
UNSAVE command
OPEN statement

CHAIN statement
OVERLAY statement
KILL statement
NAME TO statement
CLOSE statement for output file
Log-on procedure (HELLO feature)

This may be minimized by having the PIP user only do directory listings to a disk file and then listing the file on the terminal or line printer.

SECURITY OF FILES AND PRIVATE VOLUMES

When files are purged, deleted or replaced the RT-11 directory entry is deleted but the data stored on the device is not removed. It may be possible for a nonprivileged user to access the data contained in the deleted file by opening virtual array files and not writing out any information, and then examining the contents of the file. This method only works for deleted files, not active files.

If a file contains confidential information and is to be deleted or replaced, the following procedure will zero the contents of the file. This procedure must be done before the file is deleted or replaced:

```
10 OPEN "file descriptor" AS FILE VF1, FILESIZE number of blocks
                                       in file
20 VF1(I) = Ø \ I=I+1 \ GO TO 20
RUNNH
```

When the file has been zeroed, BASIC prints:

```
?SOB AT LINE 20
READY
```

BASIC has no means of checking whether a private volume belongs to the current user; consequently, BASIC uses the same file protection system for private volumes as for public devices. A nonprivileged user cannot read another user's files on a private volume but can create new files on the device. This could fill the device or it's directory and make it unusable until a privileged user deletes the unwanted files. For complete security physical access to private volumes should be restricted and the volume should only be mounted on a device unit that has been assigned by the ASSIGN command.

SECURITY OF FILES ON CASSETTES AND MAGTAPES

There is no file protection on cassettes and magtapes. Any user can read any cassette or magtape file by means of the MODE option in the OPEN statement. It is also possible to destroy all existing data on a cassette or magtape. Consequently to keep information on cassettes or magtapes confidential, physical access to the volumes must be restricted and the device drive must be assigned before the volume is mounted.

LOG-ON PRINTOUTS

During the log-on procedure the informational message may be printed with many pauses. This is because the file NOTICE.BØØ is being printed one character at a time.

Slow printing of the informational message may occur whenever the system is busy and does not indicate an error.

MAXIMUM PROGRAM LINE SIZE

Terminals with less than 132 character widths can be used to develop BASIC programs (and enter data) whose lines are 132 characters long; the automatic carriage return/line feed generated by BASIC when a user's typing exceeds the terminal margin is not considered a terminator and is not stored in the program.

However, if a user is attempting to minimize program size by maximizing the use of multiple statement lines, a BASIC program can be created whose lines are very close to the maximum 132 character length. In this situation, when the program is saved and later restored, there is a possibility that the ?LTL (Line Too Long) error message will be produced. This may occur if a line as typed in by the user did not contain the blanks that are generated by BASIC when it lists or saves a program, e.g., the blanks following keywords, and the blanks preceding and following backslashes. To prevent this from happening, the user should type in program lines in a format as close as possible to the way BASIC will list them. In this way, if the line is too long, it will be detected upon initial entry.

If a file is created that contains lines longer than 132 characters, it is impossible to read the entire file with either the OLD command or the INPUT statement; the operation stops at the line generating the error. The following program copies all lines of a file except those which are longer than 132 characters - these lines are printed on the terminal.

```
10 REM     PROGRAM TO COPY FILES ELIMINATING LINES
20 REM     LONGER THAN 132 CHARACTERS
30 DIM A(133)
40 PRINT "INPUT FILE"; \ INPUT I$
50 OPEN I$ FOR INPUT AS FILE #1
60 PRINT "OUTPUT FILE"; \ INPUT O$
70 OPEN O$ FOR OUTPUT AS FILE #2
80 PRINT
90 PRINT "LINES LONGER THAN 132 CHARACTERS:"
100 REM GET A NEW LINE
110 FOR I=0 TO 133
120 A(I)=SYS(4,1)
125 IF A(I)=-1 THEN 200 \ REM END OF FILE
127 IF A(I)=0 THEN 200 \ REM ALSO EOF
130 IF A(I)=10 THEN 500 \ REM END OF LINE
140 NEXT I
200 REM END OF LINE NOT REACHED AFTER 132 CHARACTERS
210 REM OR END OF FILE REACHED
220 FOR I=0 TO 133
230 IF A(I)=-1 THEN 600 \ IF A(I)=0 THEN 600 \ REM EOF
240 PRINT CHR$(A(I)); \ REM PRINT OUT STORED CHARACTERS
250 NEXT I
260 REM THEN PRINT ALL CHARACTERS UNTIL LINE FEED
270 A=SYS(4,1)
280 IF A=-1 THEN 600 \ IF A=0 THEN 600 \ REM EOF
290 PRINT CHR$(A);
300 IF A=10 THEN 100 \ REM LINE FINISHED
310 GO TO 270
500 REM PRINT LINE TO OUTPUT FILE
510 FOR J=0 TO I
520 PRINT #2,CHR$(A(J));
530 NEXT J
540 GO TO 100 \ REM LINE FINISHED
600 REM CLOSE ALL FILES
605 CLOSE #1,#2
610 END
```

DATE CHANGE AT MIDNIGHT

When BASIC is running under the RT-11 F/B Monitor the date is changed at midnight. For this feature to operate correctly, whenever the

date is entered by the RT-11 DATE command the correct time should be entered by RT-11 TIME command. For example, in response to the monitor's dot,

```
Type:      DAT 27-MAY-75<CR>
Response:  .

Type:      TIM 13:30:25<CR>
Response:  .
```

When BASIC is run under the SJ monitor the date returned by the DAT$ function and printed in the program header line is the last date entered by the RT-11 DATE command.

LOW-SPEED PAPER TAPE READER OPERATION

The low-speed paper tape reader (on ASR33 and ASR35 terminals) operates only after a character is typed at the keyboard. This occurs either when BASIC programs or data is being read. RUBOUT is a recommended key to type.

FILE MAINTENANCE

Certain BASIC error messages indicate that an RT-11 directory device (disk or DECtape) may have an inefficient file arrangement. These error messages are ?NER (Not Enough Room), ?DRO (DiRectory Overflow), and, when caused by a SAVE or REPLACE, ?FTS (File Too Short). If this situation occurs, exit from BASIC, run the RT-11 PIP program and "squish" the device. No foreground job may be present during this operation. In response to the monitor's dot,

```
Type:      R PIP<CR>
Response:  *

Type:      device:/S<CR>
```

where device: represents an RT-11 device unit (RK0, SY, etc.).

```
Response:  ?REBOOT?     - printed only when squishing the system
           *              device
```

If the device is the system device, reboot the system

```
Type:      SY:/O<CR>
Response:  RT-11xx Vxx-xx
```

NUMBER OF RT-11 DIRECTORY SEGMENTS

Applications in which many relatively small files are created often fill up disk directories while there are still free areas on the disk. To avoid this specify extra directory segments when zeroing the disk.

For example, in response to the monitor's dot,

```
Type:      R PIP<CR>
Response:  *

Type:      device:/Z/N!31<CR>
Response:  device:/Z ARE YOU SURE?

Type:      Y<CR>
Response:  *
```

Initializes a disk with the maximum number of directory segments, 31 (decimal). This operation must be done before an RT-11 system or any data files are transferred to the disk because the initialization destroys any previous data on the disk. See the RT-11 System Reference Manual for more information on PIP.

POWERFAIL WHEN MU BASIC/RT-11 IS RUNNING

If a powerfail occurs while BASIC is running, when power is restored the system will halt. Neither the user programs, nor the BASIC system, nor the RT-11 Monitor currently in memory is recoverable. The system must be rebooted (see Step 1 in Section 1.3.3).

LOWER CASE SUPPORT

In the distributed versions of MU BASIC/RT-11, lower case letters typed at a terminal keyboard are converted (mapped) to upper case. (Lower case can always be output using the CHR$ function, and lower case characters in a data file are always transmitted unmapped.) In certain applications, however, it may be desirable to turn off this mapping to allow direct input of lower case. The following patches will turn off lower case mapping; if they are applied, the following should be noted:

All variable names and BASIC keywords (for commands, statements, functions, etc.) must be entered in upper case; lower case is permissible only in REM statements, string constants and in response to an INPUT statement or SYS(4) function.

These patches will affect all terminals. Most terminals that are capable of generating lower case codes have a switch located on or near the keyboard that controls whether or not lower case is automatically mapped to upper case by the terminal. On terminals on which it is desired that lower case not be transmitted, this switch should be set to the "upper case only" (64 character set) position.

The first patch can be used only on the supplied MUBAS.SAV file (the underlined portions are typed by the user):

```
.R PATCH<CR>

PATCH Vxx-xx

FILE NAME--
*MUBAS.SAV<CR>
*15430/  40  0<CR>
*E
.
```

The MUBAS.REL file cannot be patched as above. If lower case input is desired while BASIC is running in the foreground, make the following patch, and relink BASIC as described in Chapter 4.

If BASIC is relinked by the user, the following patch will make the change to eliminate lower case mapping permanent:

```
.R PATCHO<CR>
*OPEN<CR>
ENTER INPUT FILE *MUBS2.OBJ<CR>
ENTER OUTPUT FILE *MUBS2.OBJ<CR>
*WORD MUBS2+1416=#0<CR>
*EXIT<CR>
ENTER CHECKSUM:   123700<CR>

STOP --
.
```

PATCH FOR CORRECT CASSETTE OPERATION

This patch fixes the problem associated with multiple OPEN files on a cassette unit. This patch MUST be done on all systems that include cassettes. After BASIC has been patched, a ?DNE error message is produced whenever a user attempts to open a second file on a cassette unit. The patched version of BASIC is MU BASIC/RT-11 V01-01A.

The first patch can be used only on the supplied MUBAS.SAV file. The underlined portions are typed by the user. <CR> indicates the RETURN key. <LF> indicates the LINE FEED key.

```
.R PATCH<CR>

PATCH Vxx-xx

FILE NAME--
*MUBAS.SAV<CR>
*1472\   4Ø          1Ø1<CR>
*5Ø26;ØR
*Ø,1Ø4Ø/             22714    4767<LF>
Ø,1Ø42/ 521Ø3        177622<LF>
Ø,1Ø44/ 14Ø3         1Ø3ØØ5<LF>
Ø,1Ø46/ 4767         12665<LF>
Ø,1Ø5Ø/ 1776Ø4       41Ø<LF>
Ø,1Ø52/ 4Ø2          44Ø<CR>
*E
.
```

The MUBAS.REL file cannot be patched as above. If cassettes are to be used when BASIC is in the foreground, make the following patch and then relink BASIC as described in Chapter 4.

If BASIC is relinked by the user, the following patches correct the problem with cassette operation and change the version number in the object modules.

```
.R PATCHO<CR>
*OPEN<CR>
ENTER INPUT  FILE *MUBS1.OBJ<CR>
ENTER OUTPUT FILE *MUBS1.OBJ<CR>
*WORD MUBS1+1Ø4Ø=#4767<CR>
*WORD +1Ø42=#-156<CR>
*WORD +1Ø44=#-74773<CR>
*WORD +1Ø46=#12665<CR>
*WORD +1Ø5Ø=#41Ø<CR>
*WORD +1Ø52=#44Ø<CR>
*EXIT<CR>
ENTER CHECKSUM:  6724Ø<CR>

STOP --
```

```
.R PATCHO<CR>
*OPEN<CR>
ENTER INPUT  FILE *MUBR.OBJ<CR>
ENTER OUTPUT FILE *MUBR.OBJ<CR>
*BYTE BASICR+6=#1Ø1<CR>
*EXIT<CR>
ENTER CHECKSUM:  1Ø5651<CR>

STOP --
.
```

CHAPTER 2

INITIAL DIALOGUE

Every time BASIC is run, a once-only initial dialogue occurs. During this dialogue several operating characteristics of the system are determined. This chapter describes the options available to the system manager and provides guidelines for making decisions.

The system manager determines the total number of users, the memory allocated to each, the optional functions to be included, and the system resources allocated to I/O. Each feature affects the total system performance; consequently, the system workload and performance should be carefully observed and changes made to the initial dialogue when appropriate. The system manager should pay special attention to error messages indicating insufficient memory for program, buffer, or device handler storage or indicating lack of I/O channels. Other questions of concern are: Is system response time adequate? Are the optional features included being used? Are any features that have been excluded needed? and, Are all the active terminals being used? When it is not clear which response in the initial dialogue will result in an optimal system, several variations should be used and the system performance observed.

BASIC has an option that allows the initial dialogue to be completed automatically. First it is necessary to complete the initial dialogue and save the responses in a configuration file. When BASIC is run subsequently, the configuration file can be specified and the initial dialogue will use the saved responses. This decreases the time needed to load the system and minimizes the operator responses.

The initial dialogue always occurs on the console terminal even if BASIC does not use the console. A CTRL/C typed at the console at any

time prior to the completion of the initial dialogue returns control to the RT-11 Monitor. To restart the initial dialogue after CTRL/C interrupts it, BASIC is restarted by the RT-11 RUN or FRUN command.

The following step-by-step procedure describes the initial dialogue.

NOTE

Except in the cases where a specific error message is described in the text, BASIC responds to an illegal initial dialogue entry by repeating the question.

STEP 1

BASIC is first run by the RT-11 RUN or FRUN command (see section 1.3.3). BASIC then checks to see if the date has been set by the RT-11 DATE command and if a real-time clock (KW11-L or equivalent) is present on the system.

If the date has not been set, BASIC prints:

```
WARNING:    DATE NOT SET
```

If this message occurs, return control to the monitor with a CTRL/C command, enter the current date, and reload BASIC.

If a clock is not present, BASIC prints:

```
WARNING:    NO CLOCK PRESENT
```

If this message appears refer to section 1.1.3 for a note concerning the clock.

After checking the date and clock (and printing a message if either is missing) BASIC prints:

```
MU BASIC/RT-11 Vxx-xx
CONFIGURATION FILE:  OLD (O), NEW (N), OR NONE (<CR>)?
```

To have BASIC automatically complete the initial dialogue from a previously created configuration file, go to Step 2.

To create a new configuration file during this dialogue, go to Step 3.

To answer the initial dialogue and not create a configuration file,

Type: <CR>

Go to Step 4.

STEP 2

To have BASIC automatically complete the initial dialogue,

Type: O<CR>

BASIC then prints:

FILE NAME --

Type: filename <CR>

where: filename is a name specified when the configuration file was created (see Step 3).

The file on the system device with the filename specified and the extension ".DØØ" is used to configure the system. No extension or device should be included with the filename. If a device or extension is specified, BASIC prints

ILLEGAL RESPONSE

and the question is repeated. If this message is printed, return control to the monitor by typing CTRL/C. BASIC must then be reloaded by the RUN or FRUN command.

If the filename specified is in the correct format but there is no file with that filename and the extension ".DØØ" on the system device, BASIC prints:

FILE NOT FOUND
FILE NAME --

If this message occurs, reenter the correct filename.

If any other error message is produced while the initial dialogue is being answered automatically, it is a fatal error. In that case type CTRL/C, rerun BASIC, and answer initial dialogue without a configuration file.

Go to Step 11.

STEP 3

To create a new configuration file,

Type: N<CR>

BASIC then prints:

FILE NAME --

Type: filename<CR>

where: filename is any valid RT-11 filename (up to six characters).

The file created is stored on the system device with the filename specified and the extension ".DØØ". For example, if the filename typed is "2USERS", the complete RT-11 file descriptor of the configuration file is "SY:2USERS.DØØ". If any device or any extension is entered, BASIC prints:

ILLEGAL RESPONSE

If this message occurs, return control to the monitor by typing CTRL/C. BASIC must then be reloaded with the RUN or FRUN command.

The file created will contain the information that is printed on the terminal during the initialization. Both the initial dialogue questions and the user's answers are stored in the file.

Any previous configuration file with the same filename will be deleted at the conclusion of the initialization. If the initial dialogue is aborted by the CTRL/C command or by a fatal error, no new configuration file is created, and any previous configuration file with the same name is still available.

If the directory of the system device is filled or if there is not enough free room on the system device for the configuration file, BASIC prints an error message and returns to the monitor. BASIC prints:

```
SYSTEM DEVICE FULL
```

At this point it is necessary to run the RT-11 PIP program and compress the system device with the /S switch or delete unwanted files with the /D switch. See the RT-11 System Reference Manual for information about the PIP program.

If a hardware write error occurs during the initial dialogue BASIC/RT-11 prints:

```
WRITE ERROR
```

and returns control to the RT-11 Monitor. If this occurs it is necessary to reload MU BASIC/RT-11 and retry the initial dialogue.

When the initial dialogue is correct up to this point, go to Step 4.

STEP 4

BASIC has printed:

```
KEEP USR RESIDENT (Y OR N)?
```

"USR" is an abbreviation for the RT-11 Monitor User Service Routine which is necessary for certain file operations. USR is needed whenever a file is opened, an output file is closed, or a BASIC program is brought into memory from a device or is saved on a device from memory. For a complete description of USR see the RT-11 System Reference Manual. For most systems, declare USR to be nonresident; this allows more memory for user program storage. To improve the performance of programs doing many file openings at a cost of approximately 2000 words of user program storage, declare USR resident.

To declare USR not resident

```
Type:     N<CR>
```

and go to Step 5.

If the available memory contains 2000 words that are not needed for user programs doing many file openings, then declare the USR resident:

```
Type:     Y<CR>
```

and go to Step 5.

If anything else is typed, the question is repeated.

STEP 5

BASIC has printed:

```
ENTER AVAILABLE DEVICES:
?
```

All devices that are to be accessed by BASIC programs must be specified at this time. Each different type of device must be entered on a different line but all units of any one device must be specified on the same line.

The general format for specifying a device is:

device[unit number][P]$\left[\text{, unit number[P], unit number[P]...}\right]$[R][B integer]

where:

device is a 2-character device name and may be:

RK	RKØ5 disk
RF	RF11 disk
DT	DECtape
CT	Cassette
MT	Magtape
PP	High-speed paper tape punch
PR	High-speed paper tape reader
CR	Card reader
LP	Line printer

or is the physical name for any other RT-11 supported device.

unit number must be an integer (between Ø and 7) and refers to a particular device drive (for example, DTØ, RK1).

P indicates that the unit is a public device. This allows more than one user to access it simultaneously. Only RT-11 directory devices (such as RK, RF, and DT) can be public.

R	causes the device handler to be resident. R should be specified unless BASIC is running in the foreground or the handler is to be loaded into the system I/O area.
B integer	causes the number of words specified to be the default buffer size for the device (must be nonfile-structured; e.g., LP, PP, PR, or CR).

To terminate the list type the Return key only.

For example, when the RT-11 devices are one RK-11 drive and a high-speed reader/punch,

```
Type:       RKØP<CR>
Response:   ?

Type:       PPR<CR>
Response:   ?

Type:       PRR<CR>
Response:   ?

Type:       <CR>
```

Or, for another example, when the RT-11 devices include two disk drives, two cassettes, and a line printer and disk drive 1 is to be a nonpublic device,

```
Type:       RKØP, 1<CR>
Response:   ?

Type:       CTØ, 1R<CR>
Response:   ?

Type:       LPR<CR>
Response:   ?

Type:       <CR>
```

If the device list is terminated with a <CR> only before the system's device has been specified, BASIC prints:

```
SYSTEM'S DEVICE NOT SPECIFIED
?
```

and requests more input devices.

If the device specification entered is not in the proper format or a device has an illegal switch specification, BASIC prints:

```
ILLEGAL DEVICE/UNIT SPECIFICATION
```

and ignores the line causing the error.

The system device must be specified by its physical name. The handler for the system device is always resident in the monitor. The R switch for the system device is ignored. The system device should be declared public for most situations especially if the HELLO feature is in use.

Allowing BASIC to access additional devices reduces the amount of memory available for user programs because space must be reserved for the device handler. The device handler can be made resident either by specifying R or by loading the handler by the RT-11 LOAD command (necessary for foreground operation). Alternatively, the device handler can be loaded into the system I/O area. The system I/O area is used for any nonresident device handler currently needed or for user file buffers that do not fit in the user's area. Using the system I/O area for device handlers can allow more memory for user programs. But successful execution of programs that require nonresident device handlers or file buffers in the system I/O area can depend on the I/O activity of other users.

There is a negligible overhead for multiple units of one device.

The sizes of the RT-11 handlers are listed in Table 2-1.

Table 2-1
Size of RT-11 Device Handlers

Device	Handler Size in Words
RK	120
DT	106
RF	92
CT	988
MT	1088
LP	99
CR	350
PR	73
PP	56

These figures represent RT-11 VØ2. They may change in future releases.

After the available devices have been listed, it is necessary to declare the number and size of system buffers. System buffers are necessary to bring BASIC program into memory using the OLD, APPEND, and RUN file descriptor commands and the CHAIN and OVERLAY statements. A minimum of one 256-word buffer is required for system operation. Additional system buffers will allow more than one user to bring BASIC programs into memory simultaneously.

Table 2-2 lists the minimum buffer size for all devices. Any device can be accessed by means of a system buffer larger than the minimum. For nonfile-structured devices, a buffer size specified with the B option in the list of available devices supersedes the standard buffer size.

Table 2-2
Standard System Buffer Sizes

Device	Minimum Buffer Required
Disk, DECtape, Magtape	256 words
Cassette	64 words
Line printer, card reader, paper tape punch/reader	16 words

BASIC has printed:

```
SYSTEM BUFFER SPECIFICATION (NUMBER, SIZE(WORDS)):
?
```

Type: integer1, integer2<CR>

where:

integer1 is the number of buffers of the specified size to be created

integer2 is the size of each buffer in words.

BASIC then prints another question mark (?). Continue entering buffers in the same format. To terminate the list type the Return key only.

For example, under most conditions one 256-word buffer is enough.

Type: 1,256<CR>
Response: ?

Type: <CR>

Or, for another example, to allow programs to be input from both cassettes and the high-speed paper tape reader without restricting access to the disk, one 256-word system buffer, two 64-word system buffers, and one 16-word system buffer could be created.

Type: 1,256<CR>
Response: ?

Type: 2,64<CR>
Response: ?

Type: 1,16<CR>
Response: ?

Type: <CR>

If a buffer specification is not in the proper format or contains invalid characters, BASIC prints:

```
ILLEGAL SYSTEM BUFFER SPECIFICATION
```

and ignores the entire line.

After the buffer list has been terminated, BASIC prints:

```
xxxxx WORDS REMAINING
SYSTEM I/O AREA SIZE (WORDS)?
```

where:

xxxxx — represents the decimal number of free words of memory.

The system I/O area is used for only two functions:

as space for nonresident device handlers

as space for user file buffers

The system I/O area should be as large as the sum of the sizes of all nonresident device handlers that are expected to be used simultaneously plus three words for each handler. The sizes of the device handlers are listed in Table 2-1. The minimum size of the system I/O area is one word. This minimum size should be specified when there are no nonresident device handlers. If any nonresident device handlers are to be used the system I/O area must be at least as large as the largest of these handlers plus three words.

There is an additional 10 word overhead for each active user file buffer.

NOTE

Nonresident device handlers decrease system performance and are not recommended unless both cassette and magtape are used and they are not to be accessed simultaneously. In this case all device handlers other than cassette and magtape should be resident and the system I/O area should be 1Ø88 words.

Type: integer<CR>

where:

integer — is the size of the system I/O area and is greater than or equal to one and restricted only by the number of free words of memory remaining.

NOTE

If a device handler has been loaded by the RT-11 LOAD command, room need not be left in the system I/O area for that device handler.

Every output from memory to a device or input from a device to memory must be transferred through a "channel". BASIC attempts to allocate a channel automatically whenever a BASIC command or statement requires one. The system manager determines the maximum number of channels to be available.

The two types of channels used in BASIC are system channels and user channels. User channels are used for BASIC data files - one user channel is allocated whenever an OPEN statement is executed - and they are also used to input or output BASIC programs. Whenever execution of an OLD, APPEND, SAVE, REPLACE, or RUN command or OVERLAY or CHAIN statement causes a BASIC program to be transferred between memory and a file, BASIC attempts to do the transfer over a user channel. If no user channel is available then the transfer is done over a system channel. There must be at least one system channel.

One user channel is required for each data file opened for input or output - enough user channels should be allocated to allow the maximum number of data files likely to be simultaneously accessed by BASIC users.

There should be as many system channels as there are system buffers.

For 15 or fewer total user and system channels each channel requires 11 words of memory. For more than 15 channels each channel requires 16 words of memory. The total maximum is limited by the amount of available memory (up to an absolute maximum of 255 channels).

For installations with 16K of memory 15 channels or fewer should be specified.

BASIC has printed:

```
NUMBER OF USER CHANNELS?
```

Type: integer<CR>

where:

integer is the number of user channels specified.

BASIC then prints:

NUMBER OF SYSTEM CHANNELS?

Type: integer<CR>

where:

integer — is the number of system channels specified and is greater than or equal to one.

NOTE

The total number of system and user channels must fit in available memory and be less than or equal to 255.

It is then necessary for the system manager to specify the maximum number of channels available for an unprivileged user, thereby restricting the number of simultaneously open data files that an unprivileged user may have. This feature prevents one user from opening so many files that the other users on the system would be restricted. To ensure that the maximum number of free channels are always available for each user, divide the user channels equally among the users. That is, the number of users times the maximum number of channels for an unprivileged user should equal the total number of user channels.

To delete this feature entirely specify the value previously entered in response to the question "NUMBER OF USER CHANNELS?". In this case any user is able to have as many channels as necessary as long as there are free channels available. Privileged users can always have as many channels as are available.

BASIC has printed:

MAXIMUM NUMBER OF CHANNELS FOR UNPRIVILEGED USER?

Type: integer<CR>

where:

integer — is the maximum number of channels for an unprivileged user.

Next the system manager must determine the maximum size file that can be created on an RT-11 directory device (disk or DECtape) by an unprivileged user. This feature prevents an unprivileged user from filling all the free space on a device with one file. There is no limit on the number of files that an unprivileged user can have. A recommended maximum size is 100 blocks for applications that do not make use of larger files. To disable this feature specify a value greater than the number of blocks on the largest device.

BASIC has printed:

```
MAXIMUM NUMBER OF BLOCKS IN OPEN FOR UNPRIVILEGED USER?
```

Type: integer<CR>

where:

integer — is the maximum number of 256-word blocks in a file created by an unprivileged user.

Next it is necessary to determine the default number of blocks for OPEN on RT-11 directory devices. This sets the default file size for the OPEN FOR OUTPUT file statement. A smaller default file size improves the efficiency of the disk but requires more frequent use of the FILESIZE option in the OPEN statement.

NOTE

Files created by the SAVE and REPLACE commands are given the standard RT-11 file allocation. This allocation is also given to files opened for output with the FILESIZE Ø option (see MU BASIC/RT-11 User's Manual).

The value specified should always be less than or equal to the value specified for the question "MAXIMUM NUMBER OF BLOCKS IN OPEN FOR UNPRIVILEGED USER?".

BASIC has printed:

```
DEFAULT NUMBER OF BLOCKS FOR OPEN?
```

Type: integer<CR>

where:

integer is the default number of 256-word blocks allocated by an OPEN FOR OUTPUT statement.

Go to Step 6.

STEP 6

Several functions can optionally be included in BASIC. Including these functions provides a more powerful language, but excluding them allows more room for user programs. The SYS function is described in Chapter 3 of the MU BASIC/RT-11 User's Manual. All other functions are described in the BASIC-11 Language Reference Manual.

The sizes (in words) of the functions are listed below.

Function Name	Size in String Version	Size in No String Version
TAB	49	Same as string version
SYS	167	165
RND	47	Same as string version
ABS	31	Same as string version
SGN	27	Same as string version
BIN	28	Same as string version
OCT	32	Same as string version
LEN	21	Not present
ASC	26	Not present
CHR$	26	18
POS	100	Not present
SEG$	84	Not present
VAL	43	Not present
TRM$	34	Not present
STR$	33	Not present

The SYS function is necessary to enable CTRL/C; consequently, the SYS function should not be deleted. When MU BASIC/RT-11 is initialized CTRL/C is treated like any other character and it is not possible to interrupt execution of a program until CTRL/C is enabled by the SYS function.

BASIC has printed:

```
OPT FNS (N-NONE, A-ALL, OR I-INDIVIDUAL)?
```

There are three possible responses: A<CR> which causes all optional functions to be included with BASIC, I<CR> which causes BASIC to list out the optional functions individually and allows each to be included or excluded independently, and N<CR> which excludes all optional functions.

NOTE

The OPT FNS question should not be answered with N<CR> (all functions excluded). If I<CR> is the response the SYS function (and, if the HELLO feature is enabled, the CHR$ function) should not be excluded.

To include all optional functions,

Type: A<CR>

and then go to Step 7.

To choose functions individually,

Type: I<CR>

BASIC prints

```
Y - YES    N - NO
TAB?
```

To include the TAB function

Type: Y<CR>

or, to exclude the TAB function,

Type: N<CR>

In either case

Response: SYS?

Type: Y<CR>

Response: RND?

Continue to respond Y<CR> or N<CR> to the functions listed.

Optional Functions String Version	Optional Functions No String Version
TAB	TAB
SYS	SYS
RND	RND
ABS	ABS
SGN	SGN
BIN	BIN
OCT	OCT
LEN	ASC
ASC	CHR$
CHR$	
POS	
SEG$	
VAL	
TRM$	
STR$	

After the list of optional functions is completed go to Step 7.

STEP 7

BASIC prints the amount of memory available for user program, array, and overhead - and then requests the number of users:

```
xxxxx WORDS CORE AVAILABLE
HOW MANY USERS?
```

where:

xxxxx is the decimal number of words remaining before the memory is divided into user areas (including overhead).

Enter the maximum number of users that will be simultaneously using the system.

To configure a system for only one user

Type: 1<CR>

and then go to Step 8.

Typing <CR> only has the same effect as typing 1<CR>.

Or, to configure a system for 2 to 8 users,

Type: integer<CR>

where:

integer is a 2, 3, 4, 5, 6, 7, or 8.

Response:
xxxxx WORDS CORE AVAILABLE
nnnnn WORDS PER USER IF DIVIDED EVENLY
CORE #1?

where:

xxxxx represents the total number of words of memory available for user program, array, and string storage. This number is smaller than the previous WORDS CORE AVAILABLE message because the overhead necessary for each user has been subtracted.

nnnnn represents the number of words of memory excluding overhead that will be allocated to each user if the CORE #1 question is answered with a <CR> only.

The CORE #1? question requests the size of the first user's memory partition. (The correspondence between the user/terminal number and the physical terminal interface is established by the vector definitions (see Table 2-3).

To allocate the memory to users evenly,

Type: <CR>

This allocates nnnnn words to each user. If nnnnn is less than the minimum memory partition allowed BASIC prints:

```
TOO SMALL
xxxxx WORDS CORE AVAILABLE
HOW MANY USERS?
```

If this message is printed repeat Step 7 and specify fewer users. Alternatively, type CTRL/C and reload MU BASIC/RT-11 and reduce the memory used by the BASIC system (determined by the features in the load module), system buffers, system I/O area, channels, resident device handlers, and optional functions.

If memory has been allocated evenly and the TOO SMALL message has not been printed, go to Step 8.

To specify the amount of memory allocated to the first user,

Type: integer<CR>

where:

integer — is the decimal number of words to be allocated to each user.

If integer is greater than the remaining free memory, BASIC prints

```
NOT ENOUGH CORE
xxxxx WORDS CORE AVAILABLE
HOW MANY USERS?
```

If this message appears repeat Step 7.

If the integer specified is less than the minimum memory size allowed or does not leave enough memory for the remaining users, BASIC prints,

```
TOO SMALL
xxxxx WORDS CORE AVAILABLE
HOW MANY USERS?
```

If this message appears repeat Step 7.

If no error message appears and the system has been configured for two users, BASIC assigns the remaining memory to the second user. In this case go to Step 8.

If there are more than two users, BASIC prints:

CORE #2?

The remaining free memory may be divided evenly by typing <CR> only. Memory can be explicitly allocated to the second user by typing

[integer]<CR>

If a TOO SMALL or NOT ENOUGH MEMORY message is produced repeat Step 7.

BASIC continues to request the memory allocation for the next user until either a response of <CR> only divides the remaining memory equally or memory has been allocated for all users except one. The last user always is allocated the remaining memory. After memory allocation is completed, go to Step 8.

STEP 8

BASIC has printed:

TERM #1?

This requests the terminal type, status register address, and vector address of each terminal. These values may be defaulted to the values listed in Table 2-3.

The terminal type may be 1, 2, or 3 and specifies whether the terminal is a fast serial terminal. Table 2-4 lists the terminal type for all supported terminals.

The status register address must be an octal number which specifies the address of the status register.

The vector address must be an octal number representing the address of the terminal interrupt vector.

For the TERM #1? and subsequent TERM #n? questions

Type: [terminal type,status register address,vector address] <CR>

Table 2-3
Default Terminal Characteristics

Term #	Terminal Type	Status Register Address	Vector Address
For MUBAS.SAV (MUBVSJ.OBJ)			
1	1	177560 (console terminal)	60
2	1	176500	300
3	1	176510	310
4	1	176520	320
5	1	176530	330
6	1	176540	340
7	1	176550	350
8	1	176560	360
For MUBAS.REL (MUBVFB.OBJ)			
1	1	176500	300
2	1	176510	310
3	1	176520	320
4	1	176530	330
5	1	176540	340
6	1	176550	350
7	1	176560	360
8	1	176570	370

Table 2-4
Terminal Types

Terminal	Terminal Type
LT33 (ASR33 or KSR33)	1
LT35 (ASR35 or KSR35)	1
LA36	1
VT50	1
VT05 at less than or equal to 300 baud[1]	1
LA30-PA (parallel)	1
LA30-PD (parallel)	1
LA30-E (serial)	2
LA30-C (serial)	2
VT05 at more than 300 baud	3

[1]Baud is a measurement of the speed of character transmission.

The default values for all three parameters may be specified by typing a <CR> only. A null, 0, or space for any parameter defaults that value. A <CR> after any parameter defaults the following parameters. See Table 2-3 for default values.

If the status register address or vector address specified does not point to a valid location, BASIC prints

```
BAD I/O ADDRESS
```

and repeats the question. If this message occurs, the address specified is not correct. If this message is produced when using default values, then the addresses of the terminal interfaces in the system are not standard and must be explicitly specified.

NOTE

The absence of BAD I/O ADDRESS message does not ensure that the addresses are correct. If MU BASIC/RT-11 does not work on some terminals, check the addresses and interface type to ensure that they are standard or as specified.

For example, a 3-terminal system, consisting of a serial LA30 DECwriter as the console terminal, an LT33 terminal as the first additional terminal, and a DL11E for a dial-up line normally used with an LT33:

```
TERM #1?  2<CR>
TERM #2?  <CR>
TERM #3?  ,175610<CR>
```

For the same hardware configuration and not using the console terminal (the usual condition for foreground operation):

```
TERM #1?  <CR>
TERM #2?  ,175610<CR>
```

There are now only two users.

After having answered the TERM # question for all terminals, go to Step 9.

STEP 9

BASIC is now running at all terminals specified in Step 8.

If BASIC has been run in the foreground by the FRUN command and the console terminal is not being used by BASIC, type CTRL/B at the console and do not type CTRL/F at the console until BASIC has been terminated. If BASIC has been run in the background and the console terminal is not being used by BASIC, type CTRL/F at the console and do not type CTRL/B at the console until BASIC is terminated.

The initial state at each terminal is

Privileged user status.
CTRL/C interrupt disabled.
User ID of AS.
The program INIT.BØØ is run.

If the supplied INIT.BØØ (without the log on feature) is present, it causes the CTRL/C command to be re-enabled, and leaves the user ID of "AS" unchanged. The program causes the user areas to be erased, and BASIC then prints:

READY

on all terminals.

If the supplied INITH.BØØ (with log on feature) has been renamed to INIT.BØØ then the message

MU BASIC/RT-11 IS ON THE AIR

is printed, and causes the program HELLO.BØØ, which controls the log-on procedure, to be executed. The log-on procedure is described in the MU BASIC/RT-11 User's Manual.

NOTE

If INIT.BØØ or HELLO.BØØ is not present on the system device MU BASIC/RT-11 prints:

```
?FNF
```

indicating that the file has not been found. At this point each user should enable CTRL/C by typing:

```
A = SYS (7)     <CR>
```

Alternatively, the system manager may exit BASIC by typing:

```
A = SYS (-3)    <CR>
```

and then transfer the missing files from the software kit (see Chapter 1).

CHAPTER 3

USING ASSEMBLY LANGUAGE ROUTINES WITH BASIC

MU BASIC/RT-11 has a facility that allows experienced PDP-11 assembly language programmers to interface their own assembly language routines (also called subprograms) to BASIC. This facility permits the user to link routines with BASIC that can operate directly on special purpose peripheral devices. This chapter serves as a programming guide for the creation of user-coded assembly language routines. The user is assumed to be familiar with PDP-11 MACRO assembly language and the RT-11 Editor. For additional information on this subject, refer to the RT-11 System Reference Manual.

Any user can execute a routine by use of the CALL statement. The CALL statement is described in the BASIC-11 Language Reference Manual. No provision is made in BASIC for user routine privacy.

3.1 SYSTEM ROUTINE TABLE

A routine is accessible from the CALL statement only if it has been defined in the special System Routine Table. This table allows BASIC to associate the name of an assembly language routine with the actual location of the routine in memory. The routine table is a list containing the name and address of every routine that may be called from BASIC.

The software provided with the BASIC kit includes a null routine table MUBI.MAC. The name of user-written assembly language routines must be defined in this module.

The table consists of a series of 3-word entries. The first two words of the entry contain the ASCII characters of the routine name to be used in the CALL statement. Those names with fewer than four characters are filled by null bytes. A 4-character name has no null bytes, a 3-character name has one null byte, a 2-character name has two null bytes (or one null word), and a 1-character name has three null bytes.

The third word of the entry contains the address of the function which must also be declared a global.

The following instructions to the EDIT program of RT-11 will produce a new MUBI that contains the names of three assembly language routines (AND, OR, and REV) and their addresses (ANDFN, ORFN, and REVFN, respectively). The sample routines in section 3.2 are the three assembly language routines that would be called by AND, OR, and REV.

Enter the following commands to the RT-11 Monitor and Editor. The monitor has printed its prompt character (a dot).

```
Type:       R EDIT<CR>
Response:   *

Type:       EWMUBI.AND<ALT><ALT>
Response:   *

Type:       ERMUBI.MAC<ALT><ALT>
Response:   *

Type:       FFTBL:<ALT><ALT>
Response:   *

Type:       I<CR>
            <TAB>  .GLOBL<TAB>     ANDFN,ORFN,REVFN<CR>
            <TAB>  .ASCII<TAB>     'AND'<CR>
            <TAB>  .BYTE <TAB>     0<CR>
            <TAB>  .WORD <TAB>     ANDFN<CR>
            <TAB>  .ASCII<TAB>     'OR'<CR>
            <TAB>  .BYTE <TAB>     0,0,<CR>
            <TAB>  .WORD <TAB>     ORFN<CR>
            <TAB>  .ASCII<TAB>     'REV'<CR>
            <TAB>  .BYTE <TAB>     0<CR>
            <TAB>  .WORD <TAB>     REVFN<CR>
            <ALT>EX<ALT><ALT>
Response:   .
```

3.2 WRITING ASSEMBLY LANGUAGE ROUTINES

The user's assembly language routine must interface with the BASIC system to receive its arguments from and return its results to the calling BASIC program.

If the user's routine accepts only a constant number of arguments, then the general subroutines GETARG, STORE, and SSTORE, provided in GETARG.MAC, may be used to interface the user routines with BASIC. The routine GETARG checks the syntax of the CALL statement and the argument types. It accesses the routine arguments as specified in the CALL statement and stores references to them in a table addressed by R0.

The GETARG routine requires that argument types be specified and that the correct number of words be stored for each argument in a table. The appropriate values are listed below (see the example in section 3.2.1 for the correct formats):

Argument Type	Stored in Table at (R0)
0 - End of argument list	Not stored
1 - Input numeric expression	Two words, the expression value
2 - Output numeric target	Three words, used by STORE subroutine
3 - Input string expression	No words are stored in table, string pointer is returned on the stack
4 - Output string target variable	Three words, used by SSTORE subroutine

To store target variables (argument types 2 and 4), the user routine addresses the corresponding 3-word entry in the table set up by GETARG and calls the subroutine STORE for numeric target variables, and SSTORE for string target variables. The examples in section 3.2.1 show how these routines are used.

Once the user routine has called GETARG to access its arguments, it may use any registers except R5 for calculations. The routine must return via an "RTS PC" instruction, with the stack unchanged. The GETARG, STORE, and SSTORE subroutines assume that all arguments to the user routines are in the CALL statement.

In the case of a user routine that handles optional arguments, the user routine may use BASIC system subroutines EVAL, GETVAR, STOVAR, and STOSVAR to pass the arguments to and from BASIC. These routines are described in section 5.2. The general procedure for using these routines is as follows:

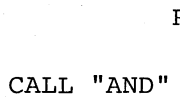
When the CALL statement is executed, the user's assembly language routine is called by the instruction:

```
JSR PC, routine address
```

When the user routine is entered, these registers contain information about the calling sequence:

R1 is a pointer to the translated code of the CALL statement. (See section 5.2 for the format of the translated code.)

If the routine has an argument list, R1 points to the 1-byte token (refer to section 5.2.4.2 for an explanation of tokens) which represents the left parentheses in the calling sequence. This token has the value .LPAR.



If the routine does not have an argument list, R1 points to a token having the value .EOL (end of line).

The 1-byte values of code bytes (tokens) .LPAR, .EOL, .COMMA and .RPAR (right parenthesis) are global symbols. These are not the same as the ASCII representation of these characters.

R4 Contains the low limit of the stack. If the stack is used heavily, the function must check that it never goes below this limit. (If it does, transfer control to ERRPDL, a global location in BASIC.)

R5 Contains the address of the "user area", which must be preserved for all calls to BASIC subroutines.

Once the argument references are no longer required by the function, R0 through R5 may be used in any way. R0, R2, and R3 need not be preserved in any case.

The routine may use the stack, but must return via an

```
RTS PC
```

instruction with the stack unchanged.

The user routine can not use the TRAP instruction, as it is reserved for use by the BASIC system program.

A user routine that does not use the GETARG subroutine should verify the syntax of the invoking CALL statement by checking that the left parenthesis, comma and right parenthesis tokens are contained in the code where expected. (.LPAR, .COMMA and .RPAR are the global values of these 1-byte tokens, respectively.)

In general, arguments that are expression values are passed to the user by the EVAL routine in BASIC. After calling EVAL with a pointer (R1) to the expression to be evaluated, the routine can obtain the value of the expression from the floating accumulator, FAC (which is FAC1(R5) and FAC2(R5)).

Arguments are passed from the user routine back to BASIC by first calling GETVAR to address the target variable and then calling STOVAR for numeric results and STOSVAR for string results to store the new value in the BASIC variable.

NOTE

Any assembly language routine linked with MU BASIC/RT-11 should not perform any RT-11 file system I/O through the RT-11 programmed requests; all RT-11 I/O should be performed at the BASIC language level. The following functions are the only RT-11 Monitor functtions that may be used safely by an assembly language routine linked with MU BASIC/RT-11:

```
.RCVDC/.RCVD
.SDATC/.SDVD
.CMKT
.CNTXSW   (a user specified list should in-
          clude EAE addresses if the EAE is
          to be used)
.DATE
.DEVICE
.GTIM
.GTJB
.MRKT     (id -1 reserved for BASIC)
.PROTECT
```

.QSET (should be used if .SDATC or .MRKT are used)
.TRPSET
.INTEN
.SYNCH

See the RT-11 System Reference Manual for a description of the function of these programmed requests.

3.2.1 Sample User Routines

The following source listing shows how the routines AND, OR, and REV in the routine table created by the editing instructions in section 3.1 would interface with the BASIC system to pass their arguments to the calling program. These routines use the general subroutines GETARG, STORE, and SSTORE in GETARG.MAC.

These routines can be executed from the BASIC program by the following implied CALL statements:

AND (A,B,C)	Calls the routine AND, which logically ANDs the binary floating point representation of A and B to produce a floating point number C. This routine uses floating point, not integer, values.
OR (A,B,C)	Calls the routine OR, which logically ORs the floating point numbers A and B to produce a floating point number, C.
REV (A$,B$)	Calls the routine REV which sets the string B$ equal to the string A$ with the characters in reverse order.

```
; FUN2 - SAMPLE USER FUNCTIONS
        .TITLE  FUN2
        .GLOBL  ANDFN, ORFN, REVFN
        .GLOBL  GETARG, STORE, SSTORE
R0=%0
R1=%1
R2=%2
R3=%3
R4=%4
R5=%5
SP=%6
PC=%7
FAC1=40
FAC2=42
;
;
;"AND"  (A,B,C)
```

```
ANDFN:  MOV     #TABLE,R0       ;ADDRESS VARIABLE STORAGE AREA
        JSR     PC,GETARG       ;CHECK SYNTAX AND SET ARGS
        .BYTE   1,1,2,0         ;(ARG TYPES)
        .EVEN
        MOV     #FAC1,R3
        ADD     R5,R3   ;ADDRESS FAC1(R5) IN R3
        MOV     A1,R2
        COM     R2
        MOV     B1,(R3)
        BIC     R2,(R3)+        ;FAC1(R5) IS A1 (AND) B1
        MOV     A2,R2
        COM     R2
        MOV     B2,(R3)
        BIC     R2,(R3) ;FAC2(R5) IS A2 (AND) B2
        MOV     #C,R0   ;ADDRESS C
        JSR     PC,STORE        ;STORE FAC1,FAC2 IN C
        RTS     PC
; "OR"I(A,B,C)
ORFN:   MOV     #TABLE,R0       ;ADDRESS ARGUMENT TABLE
        JSR     PC,GETARG       ;CHECK SYNTAX AND GET ARGS
        .BYTE   1,1,2,0         ;(ARG TYPES)
        .EVEN
        MOV     #FAC1,R3        ;ADDRESS FAC1(R5) IN R3
        ADD     R5,R3
        MOV     A1,(R3)
        BIS     B1,(R3)+        ;FAC1(R5) IS A1 (OR) B1
        MOV     A2,(R3)
        BIS     B2,(R3) ;FAC2(R5) IS A2 (OR) B2
        MOV     #C,R0   ;ADDRESS C
        JSR     PC,STORE        ;STORE FAC1,FAC2 IN C
        RTS     PC
; "REV" (A$,B$)
REVFN:  MOV     #TABLE,R0       ;ADDRESS ARG AREA
        JSR     PC,GETARG       ;CHECK SYNTAX AND GET ARGS
        .BYTE   3,4,0   ;(ARG TYPES)
        .EVEN
        CMP     (SP),#-1        ;CHECK NULL STRING
        BEQ     REVX
        CLR     R2
        MOV     (SP),R3
        BISB    (R3)+,R2        ;R2 IS STRING LENGTH
        CMPB    (R3)+,(R3)+     ;R3 ADDRESSES CHARS
REV1:   DEC     R2      ;TO SWITCH
        MOV     R3,R0
        ADD     R2,R0
        CMP     R0,R3   ;CHECK DONE--REACHED MIDDLE
        BLOS    REVX
        MOVB    (R0),R1 ;EXCHANGE ANOTHER PAIR
        MOVB    (R3),(R0)       ;OF BYTES
        MOVB    R1,(R3)+
        DEC     R2
        BR      REV1

REVX:   MOV     #B$,R0  ;ADDRESS B$
        JSR     PC,SSTORE       ;STORE STRING ON STACK
        RTS     PC
;
; ARGUMENT AREA
```

```
TABLE:
A1:     .WORD   0       ;VALUE OF A (2 WORDS)
A2:     .WORD   0
B1:     .WORD   0       ;VALUE OF B (2 WORDS)
B2:     .WORD   0
C:      .WORD   0,0,0   ;ADDRESS OF C (3 WORDS)
;
.=TABLE
                        ;POINTER TO A$ IS ON STACK
B$:     .WORD   0,0,0   ;ADDRESS OF B$ (3 WORDS)
;
        .END
```

3.3 BACKGROUND ASSEMBLY LANGUAGE ROUTINES

BASIC provides for the execution of a background assembly language routine during its idle-time. A background routine is linked with BASIC and is executed whenever BASIC is running and all users are waiting for input or output completion. The background routine is internal to BASIC and should not be confused with the foreground/background monitor. The address of the background routine must be a global and must be included in the MUBI module. For example, a routine to refresh a CRT scope display could be linked as a background routine.

NOTE

A background routine is executed only when all users on the system are I/O bound. This may not execute the background routine frequently enough for some applications.

If both a background routine and assembly language routines are linked with BASIC, they must both be included in MUBI.

To include a background routine named BKG in the MUBI module, follow this procedure:

RT-11 has printed its prompt character . (dot).

Type: R EDIT<CR>
Response: *

```
Type:           EWMUBI.BKG<ALT><ALT>
Response:       *

Type:           ERMUBI.MAC<ALT><ALT>
Response:       *

Type:           FBKGI:<ALT>I<CR>
                <TAB>    .GLOBL<TAB>    BKG<CR>
                <TAB>    .WORD <TAB>    BKG<CR>
                <ALT>EX<ALT><ALT>
Response:       .
```

The background source file should be in the following format.

```
      R0=%0
      R1=%1
      R2=%2
      R3=%3
      R4=%4
      R5=%5
      SP=%6
      PC=%7
      .GLOBL  BKG

BKG:  ;START OF BACKGROUND ROUTINE
      ;...
      ;...
      ;...
      RTS     PC

      .END
```

3.4 ASSEMBLING USER-WRITTEN ROUTINES

Any assembly language routine or background routine must be assembled and linked with MU BASIC/RT-11 before it can be accessed via the CALL statement or executed as an MU BASIC/RT-11 background routine. The instructions for assembling these files, the GETARG general subroutines, and MUBI system routine table are provided in this section. The linking instructions are provided in Chapter 4.

User-written assembly language routines and background routines must be assembled and linked by the user; however, FUN2.MAC (sample assembly language routine) and BKG.MAC (sample background routine) and the instructions for assembling and linking these routines into MU BASIC/RT-11 are examples which may be followed in creating, assembling, and linking user-written routines.

Assembling the System Routine Table

To assemble the file MUBI.AND created in section 3.2, follow this procedure:

RT-11 has printed its prompt character (.);

```
Type:          R MACRO<CR>
Response:      *

Type:          MUBI,MUBI=MUBI.AND<CR>
Response:      ERRORS DETECTED:  0
               FREE CORE:  xxxxx. WORDS
               *
```

This creates a file (MUBI.OBJ) which should be linked with MU BASIC/RT-11 in place of the MUBI.OBJ file provided with the software. MUBI.LST, a file containing a source listing, is also produced.

```
Type:          <CTRL/C>
Response:      ↑C
               .
```

Assembling the Sample Routine

This procedure assumes that the code listed in Section 3.2.1 has been inserted (with RT-11 EDIT) into a file named FUN2.MAC. Enter the following commands after RT-11 has printed its prompt character (.);

```
Type:          R MACRO<CR>
Response:      *

Type:          FUN2,FUN2=FUN2<CR>
Response:      ERRORS DETECTED:  0
               FREE CORE:  xxxxx. WORDS
               *
```

This produces the file FUN2.OBJ which should be linked with MU BASIC/RT-11 as described in Chapter 4.

```
Type:          <CTRL/C>
Response:      ↑C
               .
```

Assembling the General Interface Subroutines

GETARG.MAC may be assembled into two object modules GET.OBJ (to be linked with MU BASIC/RT-11 which supports string features) and GETNS.OBJ (no string features). The following instructions will produce both files.

RT-11 has printed its prompt character (.);

```
Type:          R MACRO<CR>
Response:      *

Type:          GET,GET=GETARG<CR>
Response:      ERRORS DETECTED:  0
               FREE CORE:  xxxxx. WORDS
               *

Type:          <CTRL/C>
Response:      ↑C
               .

Type:          R EDIT<CR>
Response:      *

Type:          EWGETNS.MAC<ALT><ALT>
Response:      *

Type:          ERGETARG.MAC<ALT><ALT>
Response:      *

Type:          F;$NOSTR<ALT>0AD<ALT><ALT>
Response:      *

Type:          EX<ALT><ALT>
Response:      .

Type:          R MACRO<CR>
Response:      *

Type:          GETNS,GETNS=GETNS
Response:      ERRORS DETECTED:  0
               FREE CORE:  xxxxx. WORDS
               *
```

```
Type:          <CTRL/C>
Response:      ↑C
               .
```

Assembling the Example Background Routine

This procedure assumes that the background routine listed in section 3.3 is present as file BKG.MAC.

After RT-11 has printed its prompt character (.);

```
Type:          R MACRO<CR>
Response:      *

Type:          BKG,BKG=BKG
Response:      ERRORS DETECTED:  0
               FREE CORE:  xxxxx. WORDS
               *

Type:          <CTRL/C>
Response:      ↑C
               .
```

CHAPTER 4

LINKING MU BASIC/RT-11

This chapter describes the procedures for linking BASIC. There are several options available when BASIC is linked. These options are used to:

Overlay the BASIC system.
Support string features.
Support special arithmetic hardware.
Include the transcendental functions.
Include the PRINT USING statement processor.
Include assembly language routines and the CALL statement processor.
Include a background routine.
Link BASIC to run in either the background (and SJ Monitor) or foreground.
Use the console terminal.

The assembly language routine and background routine features are described in Chapter 3. All other options are discussed in Section 1.3.1.

Follow the step-by-step linking procedure in Section 4.1 after deciding which optional features are to be included. Section 4.2 contains some demonstration linkings.

4.1 STEP-BY-STEP LINKING INSTRUCTIONS

In these instructions, square brackets indicate optional elements of the filenames. Do not type the brackets. If string features are to be included, do not type any of the letters within the square brackets. If string features are to be excluded, type all the letters within the brackets.

STEP 1

If BASIC is not to include support for special arithmetic hardware, go to Step 2.

The RT-11 Monitor has printed its prompt character (.);

```
Type:          R PIP<CR>
Response:      *
Type:          MUBM1.BAR=MUBM1.OBJ/R<CR>
Response:      *
Type:          MUBM2.BAR=MUBM2.OBJ/R<CR>
```

If there is a PDP-11/45 FPU hardware arithmetic processor,

```
Type:          MUBM1.OBJ=MUBM1.FPU/R<CR>
Response:      *
Type:          MUBM2.OBJ=MUBM2.FPU/R<CR>
Response:      *
```

Or if there is an EAE arithmetic hardware processor,

```
Type:          MUBM1.OBJ=MUBM1.EAE/R<CR>
Response:      *
Type:          MUBM2.OBJ=MUBM2.EAE/R<CR>
Response:      *
```

Or if there is an EIS arithmetic hardware processor,

```
Type:          MUBM1.OBJ=MUBM1.EIS/R<CR>
Response:      *
Type:          MUBM2.OBJ=MUBM2.EIS/R<CR>
Response:      *
```

Or if there is an FIS,

```
Type:          MUBM1.OBJ=MUBM1.FIS/R<CR>
Response:      *
Type:          MUBM2.OBJ=MUBM2.FIS/R<CR>
Response:      *
```

In any case,

```
Type:       <CTRL/C>
Response:   ↑C
            .
```

Go to Step 2.

STEP 2

The RT-11 Monitor has printed its prompting (.);

```
Type:       R LINK<CR>
Response:   *
```

To link BASIC to run as a Foreground job go to Step 3.

To link MU BASIC/RT-11 to run under the SJ monitor or as a background job:

```
Type:       MUBAS[N],MUBAS[N]=MUBA/B:400/C<CR>
Response:   *
```

Go to Step 4.

STEP 3

To link MU BASIC/RT-11 to run as a foreground job:

```
Type:       MUBAS[N],MUBRE[N]=/R/C<CR>
Response:   *
```

STEP 4

To exclude the console terminal go to Step 5 (recommended for foreground operation).

To use the console terminal,

```
Type:       MUBVSJ/C<CR>
Response:   *
```

Go to Step 6.

STEP 5

To exclude the console terminal,

```
Type:       MUBVFB/C<CR>
Response:   *
```

Go to Step 6.

STEP 6

```
Type:       MUBR[NS],MUBS1,MUBS2,MUBT,MUBI/C<CR>
Response:   *
```

To exclude assembly language routines, go to Step 7.

To include assembly language routines, link them at this point. These instructions link the sample assembly language routines in the file FUN2.OBJ (see Chapter 3). It is important that the MUBI.OBJ in which the assembly language routines are defined be used in Step 6 instead of the file MUBI.OBJ provided with BASIC. Include the general argument subroutine GET[NS] (assembled as shown in Chapter 3) if the assembly language routine uses it.

```
Type:       FUN2,GET[NS]/C<CR>
Response:   *
```

Go to Step 8.

STEP 7

To exclude assembly language routines and the CALL statement processor,

```
Type:       MUBNC/C<CR>
Response:   *
```

Go to Step 8.

STEP 8

To exclude a background routine go to Step 9.

To include a background routine link it at this point. These instructions link the background routine in file BKG created by the procedure given in Chapter 3. The MUBI.OBJ linked in Step 6 must be produced by the procedure in Chapter 3 and cannot be the MUBI.OBJ provided with BASIC.

```
Type:       BKG/C<CR>
Response:   *
```

Go to Step 9.

STEP 9

To exclude the PRINT USING statement processor, go to Step 10.

To include the PRINT USING statement processor,

```
Type:       MUBPR/C<CR>
Response:   *
```

Go to Step 11.

STEP 10

To exclude the PRINT USING statement processor,

```
Type:       MUBNP/C<CR>
Response:   *
```

Go to Step 11.

STEP 11

If the transcendental functions are to be excluded go to Step 12.

If the transcendental functions are to be included,

```
Type:       MUBM1,MUBM2/C<CR>
Response:   *
```

Go to Step 13.

STEP 12

To exclude the transcendental functions,

```
Type:       MUBM1,MUBNM2/C<CR>
Response:   *
```

Go to Step 13.

STEP 13

If the BASIC system is not to use overlays, go to Step 14.

To create an overlaying BASIC system,

```
Type:       MUBE[NS],MUBS2E,MUBET/0:1/C<CR>
Response:   *
```

Go to Step 15.

STEP 14

To create a nonoverlaying BASIC system,

```
Type:       MUBE[NS],MUBS2E,MUBET/C<CR>
Response:   *
```

Go to Step 15.

STEP 15

```
Type:       MUBX[NS]/0:1/C<CR>
Response:   *
```

If the PRINT USING statement processor has been excluded (in Step 10), go to Step 16.

If the PRINT USING statement processor has been included (in Step 9),

Type: MUBPX[N]/C<CR>
Response: *

Go to Step 16.

STEP 16

If assembly language routines and the CALL statement processor have been excluded (in Step 7), go to Step 17.

If assembly language routines have been included (in Step 6),

Type: MUBC/C<CR>
Response: *

Go to Step 17.

STEP 17

Type: MUBXT/C<CR>
Response: *

Type: MUBO/0:1/C<CR>
Response: *

Type: MUBH[NS],MUBZ/0:2<CR>

At this point the actual linking occurs. The file MUBAS[N].MAP (or if linked for the foreground, MUBRE[N].MAP) is produced. This contains the link map. See the RT-11 System Reference Manual for more information about the link map. If BASIC is linked for the background (or the SJ Monitor), the file MUBAS[N].SAV is produced. If BASIC is linked for the foreground, the file MUBAS[N].REL is produced.

After these files have been produced, the RT-11 Linker prints several line feeds and an asterisk (*) to indicate that it is ready to accept more commands.

4.2 LINKING EXAMPLES

This section contains the actual terminal output produced when BASIC is linked with various options. The RT-11 linker accepts up to six input files on a line; consequently, some of the continued lines (indicated by /C) of the step-by-step procedure have been combined for convenience.

Link #1 Standard BASIC for Background

These instructions re-create the MUBAS.SAV file supplied in the binary kit.

```
.R LINK
*MUBAS=MUBA,MUBVSJ/B:400/C
*MUBR,MUBS1,MUBS2,MUBT,MUBI/C
*MUBNC,MUBPR,MUBM1,MUBM2/C
*MUBE,MUBS2E,MUBET/O:1/C
*MUBX,MUBPX,MUBXT/O:1/C
*MUBO/O:1/C
*MUBH,MUBZ/O:2

*
```

Link #2 Standard BASIC for Foreground

These instructions re-create the MUBAS.REL file supplied in the binary kit.

```
.R LINK
*MUBAS=MUBVFB/R/C
*MUBR,MUBS1,MUBS2,MUBT,MUBI/C
*MUBNC,MUBPR,MUBM1,MUBM2/C
*MUBE,MUBS2E,MUBET/O:1/C
*MUBX,MUBPX,MUBXT/O:1/C
*MUBO/O:1/C
*MUBH,MUBZ/O:2

*
```

Link #3 Nonstandard BASIC for Background

These linking instructions create a nonoverlaying BASIC with support for the EAE arithmetic hardware processor, with no string features, with no PRINT USING statement processor. This BASIC will run in the background or under the SJ Monitor.

```
.R PIP
*MUBM1.BAR=MUBM1.OBJ/R
*MUBM2.BAR=MUBM2.OBJ/R
*MUBM1.OBJ=MUBM1.EAE/R
*MUBM2.OBJ=MUBM2.EAE/R
*^C

.R LINK
*MUBASN,MUBASN=MUBA,MUBVSJ/B:400/C
*MUBRNS,MUBS1,MUBS2,MUBT,MUBI/C
*MUBNC,MUBNP,MUBM1,MUBM2/C
*MUBENS,MUBS2E,MUBET/C
*MUBXNS,MUBXT/O:1/C
*MUBO/O:1/C
*MUBHNS,MUBZ/O:2


*
```

CHAPTER 5

INTERNAL DESCRIPTION AND ASSEMBLING INSTRUCTIONS

5.1 ASSEMBLING INSTRUCTIONS

MU BASIC/RT-11 is also available in source form. This section describes the procedure to re-create the object files supplied in the binary kit. In addition, the user can assemble a version of BASIC that includes longer, more descriptive error messages and can selectively omit transcendental functions. The source kit is necessary for users who wish to make alterations to the internal structure of BASIC (DIGITAL does not support any user-altered software).

The following files are provided in the source kit:

```
BASR.MAC
BASX.MAC
NOCALL.MAC
NOPRU.MAC
NOSTR.MAC
LONGER.MAC
MUBA.MAC
MUBC.MAC
MUBE.MAC
MUBET.MAC
MUBH1.MAC
MUBH2.MAC
MUBH3.MAC
MUBI.MAC
MUBNM2.MAC
MUBP.MAC
MUBP1.MAC
MUBP2.MAC
MUBR.MAC
MUBS1.MAC
MUBS2.MAC
MUBS2E.MAC
MUBT.MAC
```

```
MUBVSJ.MAC
MUBVFB.MAC
MUBX.MAC
MUBXT.MAC
MUBZ.MAC
EAE.MAC
EIS.MAC
FIS.MAC
FPU.MAC
MATH.MAC
CONV2.MAC
CONV3.MAC
ERMOD.MAC
FADD.MAC
FDIV.MAC
FMUL.MAC
INTR.MAC
ALOG.MAC
ATAN.MAC
EXP.MAC
SIN.MAC
SQRT.MAC
CT.MAC
```

To assemble these files into the object modules provided, follow the procedure described below.

The monitor has printed its prompting character (.);

```
Type:       R MACRO<CR>
Response:   *

Type:       MUBC,MUBC=MUBP1,MUBP2,MUBC<CR>
Response:   ERRORS DETECTED:  0
            FREE CORE:  xxxxx. WORDS
            *
```

This procedure assembles the files on the right side of the equal sign (MUBP1.MAC,MUBP2.MAC, and MUBC.MAC) and produces an object file (MUBC.OBJ) and a listing file MUBC.LST). For more information on the RT-11 MACRO assembler see the RT-11 System Reference Manual.

To eliminate the listing file, enter only one file to the left of the equal sign, for example:

```
Type:       MUBC=MUBP1,MUBP2,MUBC<CR>
```

To assemble a version of BASIC with longer error messages,

Type: MUBC=LONGER,MUBP1,MUBP2,MUBC<CR>

After MACRO has printed the asterisk (*) the next command can be entered. Type the following commands (the asterisk is printed by MACRO). MACRO prints the ERRORS DETECTED and FREE CORE messages in response to each command line. Include LONGER to create a version of BASIC with longer error messages.

```
*MUBA=[LONGER,]MUBA<CR>
*MUBE=[LONGER,]MUBP1,MUBP2.MUBE<CR>
*MUBENS=[LONGER,]NOSTR,MUBP1,MUBP2.MUBE<CR>
*MUBET=[LONGER,]MUBET<CR>
*MUBH=[LONGER,]MUBP1,MUBP2,MUBH1<CR>
*MUBHNS=[LONGER,]NOSTR,MUBP1,MUBP2,MUBH1<CR>
*MUBI=[LONGER,]MUBI<CR>
*MUBC=[LONGER,]MUBP1,MUBP2,MUBC
*MUBNC=[LONGER,]NOCALL,MUBP1,MUBP2,MUBC<CR>
*MUBNP=[LONGER,]NOPRU,MUBP1,MUBP2,MUBP<CR>
*MUBO=[LONGER,]MUBP1,MUBP2,MUBH2,MUBH3<CR>
*MUBPR=[LONGER,]BASR,MUBP1,MUBP2,MUBP<CR>
*MUBPX=[LONGER,]BASX,MUBP1,MUBP2,MUBP<CR>
*MUBPXN=[LONGER,]BASX,NOSTR,MUBP1,MUBP2,MUBP<CR>
*MUBR=[LONGER,]MUBP1,MUBP2,MUBR<CR>
*MUBRNS=[LONGER,]NOSTR,MUBP1,MUBP2,MUBR
*MUBS1=[LONGER,]MUBP1,MUBP2,MUBS1
*MUBS2=[LONGER,]MUBP1,MUBP2,MUBS2
*MUBS2E=[LONGER,]MUBP1,MUBP2,MUBS2E
*MUBT=[LONGER,]MUBT
*MUBVSJ=[LONGER,]MUBVSJ
*MUBVFB=[LONGER,]MUBVFB
*MUBX=[LONGER,]MUBP1,MUBP2,MUBX
*MUBXNS=[LONGER,]NOSTR,MUBP1,MUBP2,MUBX
```

```
*MUBXT=[LONGER,]MUBXT
*MUBZ=[LONGER,]MUBZ
*MUBNM2=[LONGER,]MUBNM2
```

The math package consists of several source files which are assembled individually and then combined into two object files. One contains the floating point math operations (MUBM1) and the other contains the transcendental functions (MUBM2).

To assemble the math package, follow this procedure:

```
Type:          <CTRL/C>
Response:      .

Type:          R PIP<CR>
Response:      *
```

To create a math package that uses the EAE arithmetic hardware,

```
Type:          MATH.MAC=EAE.MAC,MATH.MAC<CR>
Response:      *
```

Or to use the EIS arithmetic hardware,

```
Type:          MATH.MAC=EIS.MAC,MATH.MAC<CR>
Response:      *
```

Or to use the FIS arithmetic hardware,

```
Type:          MATH.MAC=FIS.MAC,MATH.MAC
Response:      *
```

Or to use the FPU arithmetic hardware,

```
Type:          MATH.MAC=FPU.MAC,MATH.MAC<CR>
Response:      *
```

To include longer error messages,

```
Type:          MATH.MAC=LONGER.MAC,MATH.MAC<CR>
Response:      *
```

In any case,

```
Type:           <CTRL/C>
Response        ↑C
                .

Type:           R MACRO<CR>
Response:       *

Type:           CONV2=MATH,CONV2<CR>
Response:       ERRORS DETECTED:  0
                FREE CORE:  xxxxxx. WORDS
```

Repeat this for the following command lines (the asterisk is printed by MACRO). MACRO prints the ERRORS DETECTED and FREE CORE messages for each command line entered.

```
*CONV3=MATH,CONV3<CR>
*ERM0D=MATH,ERM0D<CR>
*FADD=MATH,FADD<CR>
*FDIV=MATH,FDIV<CR>
*FMUL=MATH,FMUL<CR>
*INTR=MATH,INTR<CR>
*ALOG=MATH,ALOG<CR>
*ATAN=MATH,ATAN<CR>
*EXP=MATH,EXP<CR>
*SIN=MATH,SIN<CR>
*SQRT=MATH,SQRT<CR>
```

At this point MACRO prints

```
                *
Type:           <CTRL/C>
Response:       ↑C
                .

Type:           R PIP<CR>
Response:       *
```

```
Type:       TEMP.OBJ=CONV2.OBJ,CONV3.OBJ,ERMOD.OBJ,FADD.OBJ/B
            <CR>
Response:   *

Type:       MUBM1.OBJ=TEMP.OBJ,FDIV.OBJ,FMUL.DBJ;INTR.OBJ/B<CR>
Type:       MUBM2.OBJ=ALOG.OBJ,ATAN.OBJ,EXP.OBJ,SIN.OBJ,SQRT.
            OBJ/B<CR>
Response:   *
```

At this point all the object files needed to link BASIC have been produced. See Chapter 4 for linking instructions.

To eliminate the LOG (and LOG10), TAN, EXP, SIN (and COS), or SQR function simply omit the file ALOG.OBJ, ATAN.OBJ, EXP.OBJ, SIN.OBJ, or SQRT.OBJ, respectively, in the instructions to PIP. If this is done BASIC produces several undefined global error messages when it is linked. These can be ignored. Eliminating either the LOG or EXP function also makes it impossible to evaluate the expression A↑B where B is noninteger or greater than 256.

5.2 INTERNAL DESCRIPTION OF MU BASIC/RT-11

This section provides an internal description of MU BASIC/RT-11 for users who wish to link assembly language routines that use BASIC system routines and for users who wish to alter the sources of MU BASIC/RT-11.

NOTE

DIGITAL does not support any user-altered software.

The routines described in this section may be changed in future releases. The reader is assumed to be familiar with the PDP-11 MACRO assembly language.

5.2.1 System Routines in MU BASIC/RT-11

Routine Name (Global)	Call	Description
BOMB	TRAP 0 .ASCIZ 'MESSAGE' .EVEN	This routine stops execution of the user's BASIC program and types the message: ?MESSAGE AT LINE xxxxx on the user's terminal. If the $LONGER option is specified when BASIC is assembled from the sources, the "?" character is omitted. BASIC then prints the READY message.
BKGI	MOV @#BKGI,R0 JSR PC,@R0	Address of assembly language background routine.
ERRSYN	JMP ERRSYN	Syntax error. Stops execution and prints ?SYN AT LINE xxxxx.
ERRARG	JMP ERRARG	Argument error. Stops execution and prints ?ARG AT LINE xxxxx.
EVAL	JSR PC,EVAL	Evaluate expression. R1 points to the start of the expression in the code. EVAL sets the carry bit as follows: carry = 0: The expression is numeric. The value of the expression is contained in the floating accumulator (FAC1 and FAC2). carry = 1: A string expression. If the string is non-null, the top of the stack is an indirect pointer to the string. (See section 5.23 for the format of string variables.) If the string is null, the top of the stack is the value 177777.

Routine Name (Global)	Call	Description
		In both cases, R1 is moved to point to the byte following the expression in the code. If it detects an error in the expression, EVAL branches to the appropriate error routine.
FTABI	MOV @#FTABI,R0	Contains address of the system routine table described in section 6.1. Routine table may be examined by assembly language routine to find names and addresses of other assembly language routines.

GETVAR	JSR PC,GETVAR	Address variable or array element. R2 must contain the address of the symbol table entry for the variable and R1 must point to the next byte beyond the second byte of the symbol table offset on call. GETVAR looks up and saves the address of the variable reference, so that a subsequent STOVAR or STOSVAR will store a value in the addressed variable. GETVAR destroys the FAC when addressing an array element; R1 is left unchanged unless the variable is subscripted, in which case R1 is advanced past the right parenthesis. To address the symbol table entry, precede the GETVAR call with the code:
		
		`MOVB (R1)+,R2 ;FIRST BYTE OF ;OFFSET`
		`BMI ESYN ;IF NEGATIVE, ERROR`
		`SWAB R2`
		`BISB (R1)+,R2 ;GET 2ND HALF OF ;OFFSET`
		`ADD (R5),R2 ;ADD BASE OF SYMBOL ;TABLE`

MSG	JSR R1,MSG .ASCIZ 'MESSAGE' .EVEN	Print message on user's terminal. Prints the ASCII characters specified after the JSR instruction up to the 0-byte. MSG prints only those characters specified in the calling sequence plus padding characters specific to the terminal in use. The calling program must insert a carriage return where required. MSG clears the CTRL/O condition. The instruction after 'MESSAGE' is then executed.

Routine Name (Global)	Call	Description
STOVAR	JSR PC,STOVAR	Store numeric variable. Stores the FAC in the variable or array element last referenced by GETVAR. If it was a string variable, STOVAR stops execution of the program and produces the ?NSM error message.
STOSVAR	JSR PC,STOSVAR	Store string variable. Stores the top of the stack in the variable or array element last referenced by GETVAR, and pops one word from the stack. If it was a numeric variable, STOSVAR stops execution of the program and produces the ?NSM error message:
INT	JSR PC,INT	Integerize the FAC. Sets the value of the FAC to the greatest integer contained in the previous contents of the FAC. The number is expressed in the BASIC integer format if possible.
MAKEST	JSR PC,MAKEST	Make non-null string variable. The top of the stack contains the length of the string to be created. R2 must point to a word which contains 3 less than the address of the first of the characters to fill the string. MAKEST returns an indirect pointer to the string on the top of the stack. (Called MAKESTR in sources.) To create a string "ABCD" the following code could be used:

```
        MOV #4,-(SP)        ;LENGTH
                            ;TO TOP OF
                            ;STACK
        MOV #STRING-3,R2    ;ADDRESS
        JSR PC,MAKEST       ;OF FIRST
                            ;CHAR -3
        MOV (SP)+,R2        ;R2 NOW
             .              ;CONTAINS
             .              ;ADDRESS
             .              ;OF BASIC
                            ;STRING
STRING: .ASCII /ABCD/
```

In addition, the user program may call the following FPMP routines. Their calling sequences are documented in the FPMP-11 User's Manual (DEC-11-NFPMA-A-D).

$POLISH	Enter "Polish Mode"
$IR	Integer-to-Real Conversion

$MLR	Multiply Real
$DVR	Divide Real
$ADR	Add Real
$SBR	Subtract Real
SIN	Sine Function (if linked with BASIC)
COS	Cosine Function (if linked with BASIC)
SQRT	Square Root Function (if linked with BASIC)
ALOG	Logarithm Function (base e) (if linked with BASIC)
ALOG10	Logarithm Function (base 10) (if linked with BASIC)
ATAN	Arctangent Function (if linked with BASIC)
EXP	Exponentiation Function (if linked with BASIC)

The following list contains all the .GLOBL symbols available to the user's assembly language routines in this release of BASIC.

GLOBL Symbol	Description
BKGI	Address of the background routine
BOMB	Error routine, called by TRAP 0
ERRARG	Argument error
ERRPDL	Stack overflow error
ERRSYN	Syntax error
EVAL	Evaluate expression
FTABI	Address of the system function table
GETVAR	Address variable
INT	Integerize floating accumulator
MAKEST	Create a string
MSG	Print a message on the terminal
STOSVAR	Store string variable
STOVAR	Store numeric variable
.COMMA	Comma token ,
.DQUOT	Double quote taken "
.EOL	End-line token \
.LPAR	Left-parenthesis token (
.RPAR	Right-parenthesis token)
.SQUOT	Single quote token '

The offset of system variables in the "user area", which starts at the address contained in R5, will not change from release to release; if new ones are added, they will be inserted at the end of the user's area. Therefore, these values may be set by MACRO equate statements in the user's source program (e.g., FAC1 = 40). The most commonly used user area offsets are described below.

User area offset	Description
SYMBOLS = 0	Address of symbol table
CODE = 16	Address of stored program
LINE = 20	Address of input line buffer
VARSAVE = 22	Saved symbol table entry address
SS1SAVE = 24	Saved first array subscript
SS2SAVE = 26	Saved second array subscript
FAC1 = 40	Floating accumulator, upper word
FAC2 = 42	Floating accumulator, lower word

5.2.2 Representation of Numbers in BASIC

The value stored in the floating accumulator (FAC1(R5) and FAC2(R5)) by EVAL is always two words long: FAC1(R5) contains the high-order, and FAC2(R5), the low-order portion. If FAC1(R5) is non-zero, then the number is stored as a 2-word floating-point number, in this format:

Word	Bit(s)	Description
FAC1(R5)	15	Sign bit, set if the number is negative.
	14-7	Exponent, with a bias of 200 octal.
	6-0	The second through eighth significant bits of mantissa. The first significant bit is always an assumed 1.
FAC2(R5)	15-0	The 9th through 24th significant bits of mantissa. The mantissa is expressed as an absolute magnitude - not in twos complement.

If FAC1(R5) is zero then FAC2(R5) contains the integer value of the number in 2's complement form. Note that the integers from -32,768 to +32,767 do not have a unique representation: they may be stored in the floating-point or integer form. For example, the number represented by:

```
FAC1:       40640      ;Floating-point "5"
FAC2:       0
```

has the same value as:

```
FAC1:       0
FAC2:       5          ;Integer "5"
```

The function INT, described in the BASIC-11 Language Reference Manual converts a number from the floating point representation to an integer.

5.2.3 Representation of Strings in BASIC

Non-null strings are represented as follows:

Byte(s)	Contents
0	The length of the string, N
1 and 2	An internal "back-pointer" used by BASIC. Do not change this value.
3 to 2+N	The ASCII characters of the string
3+N	The length of the string, N

A null string is not stored in BASIC; rather, the indirect pointer to the string has the value 177777.

5.2.4 Format of Translated BASIC Program

When the user enters a BASIC program, the BASIC system does not store the program exactly as it is typed or read from the input file. Instead, it translates the program to an intermediate form which can be used in two different ways. The intermediate code can be "untranslated" by the LIST or SAVE commands to produce an ASCII program which looks very similar to the input program, or the translated code can be interpreted by the RUN command to provide execution of a program.

5.2.4.1 Symbol Table Format - As the BASIC program is input, the system builds a symbol table in memory at the indirect address 0(R5). There are four different types of symbol table entries, as shown in Table 5-1.

Table 5-1

Symbol Table Entries

Symbol Table Definition	Description
Line Number	This entry is two words long: Word 1: Line number is an unsigned 15-bit integer. The highest number allowed is 77777 octal or 32,767 decimal. Word 2: The address of the specified line in the stored translated program.
Numeric Scalar	This is five words long: Word 1: Constant 177775 Word 2: High-order Scalar Value Word 3: Low-order Scalar Value Word 4: Constant 0 Word 5: ASCII scalar name, the second byte is 0 if the name is only one character.
Numeric Array	This entry is five words long: Word 1: Constant 177776 Word 2: Address of array Word 3: Maximum value of first subscript (SSlMAX below) Word 4: Maximum value of second subscript or -1 if the array is singly dimensioned. Word 5: ASCII array name. The scalar with the same name as an array is stored internally before the first element of the array. The address of the array is actually the address of this element. The arrays are stored with the first subscript varying the fastest; each element of the array takes up two words. The address of the (M,N) element in the array is the array address plus the quantity: 4*(N*SSlMAX+M+1)

Table 5-1 (Cont.)

Symbol Table Entries

Symbol Table Definition	Description
String	This entry is five words long: Word 1: Constant 177777 Word 2: Array address, or string pointer, if Word 3=-1 Word 3: Maximum value of first subscript (SSlMAX below), or -1 if not a string array Word 4: Maximum value of second subscript, or -1 if the array is singly-dimensioned or scalar Word 5: ASCII string name, with '$' character omitted Strings and string arrays are stored as 1-word pointers to the strings, or the flat 177777 for a null string. If a string is dimensioned or used as a string array, the scalar string with the same name is stored before the first entry in an array. Otherwise, the pointer to the scalar string is stored directly in the symbol table entry, as indicated above. The address of the pointer to the (M,N) element in the array is then the array address plus the quantity: 2*(N*SSlMAX+M+1)

5.2.4.2 Translated Code - After the line is input, the TRAN subroutine is called to translate it to the internal format. TRAN scans the input line from left to right, and translates it as described below.

All references to line numbers or variable names are stored as the 2-byte offset into the symbol table of the entry for that variable name. The symbol table entries for all numeric variables are initially scalars, and are changed to dimensioned arrays when the RUN statement is executed. This 2-byte offset is, of course, not negative; therefore, it may be distinguished from the "keyword tokens" described below. It is not necessarily aligned to a word boundary.

All sequences of characters used as a single unit by the BASIC language are defined as "Keywords". The following are examples of keywords:

```
LET
INPUT
STEP
+
(
)
SIN(
GO TO
RANDOMIZE
```

TRAN scans the characters in the program line for the occurrence of any of the keywords, disregarding blanks. When one is found, the corresponding 1-byte system "token" is stored in the saved program. Thus, only one byte in the stored program is required to store such keywords as GOSUB and RANDOMIZE. All of the tokens have the high-order bit set.

At the end of every line and at every backslash in the code, there is a special ".EOL" token. At the end of the program there is an ".EOF" token.

The values of the tokens may be found in a listing of BASIC. Since they are only used internally, some of the values may be different for different versions of BASIC.

When an integer literal is encountered in the program following a GOSUB, GO TO, THEN, LIST, or LISTNH keyword, or as the first element on a line, it is stored as a symbol table reference to a line number entry.

When TRAN finds any other literal numeric value in the input program line, TRAN stores the value in the translated program in one of the following forms:

1-byte literal — An integer constant in the range 0-255 is stored as two bytes in the translated program:

Byte 1: constant 375
Byte 2: 1-byte value

1-word literal — An integer constant with an absolute value less than 32,768 which is not in the range 0-255 is stored as three bytes in the translated program:

Byte 1:	constant 376
Bytes 2-3:	2-byte value

2-word literal — Any other numeric constant is stored as five bytes in the translated program:

Byte 1:	constant 374
Bytes 2-5:	4-byte floating point value of the literal, as described in section 5.2.2.

Certain keywords when translated into tokens have special "extra bytes" inserted after the token as described below.

Keyword	Translated Code
' or "	When the first quote character is encountered, TRAN outputs the corresponding token, followed by a .TEXT token, with the value 377. Next follow all of the ASCII characters in the program line up to the closing quote character. Finally, TRAN outputs a 0 byte and a matching close-quote token to the translated program.
FN	A special byte is placed in the translated code after the FN token. It contains a function number to represent the function name, as follows:

Function Number (octal)	Function Name
0	FNA
2	FNB
4	FNC
6	FND
.	.
.	.
.	.
62	FNZ

NEXT	Ten extra bytes are output to the translated code following the NEXT statement; these are required at execution time for the proper nesting of FOR-NEXT loops.
REM	The REM token in the code is followed by a .TEXT token, and then the remaining characters and then a zero byte and an .EOL token.

Any character sequence that cannot be translated into a token and is not a symbol table reference or literal, is translated as the .TEXT token, followed by the remaining characters on the line. The BASIC language does not allow a program to have two adjacent variable names

without an intervening character. If this occurs, the remainder of the line is translated as described above. When any such translated program line is executed, it produces a syntax error. If the first sequence of characters in a program statement is translated as a .TEXT token, BASIC assumes an implicit CALL statement (if the CALL statement processor has been included with BASIC). The characters are compared with the assembly language call statements defined in the system routine table. If the characters match a call name the corresponding routine is called. If a match is not found the ?MSP (Missing Sub-Program) error message is printed.

APPENDIX A

DIFFERENCES BETWEEN BASIC/RT-11 V01 AND MU BASIC/RT-11

This appendix describes the language differences between single user BASIC/RT-11 V01 and MU BASIC/RT-11. This information is for users who are upgrading to MU BASIC/RT-11 or who are concerned with compatibility with single user BASIC/RT-11 V01.

The following features are found only in MU BASIC/RT-11 and are not present in BASIC/RT-11 V01:

Statements

COMMON
PRINT USING
ON GO TO
ON GOSUB
KILL
NAME TO
RESET
LINPUT

Functions

LOG10
PI
SYS

Commands

APPEND
SET TTY
UNSAVE
ASSIGN
DEASSIGN
BYE
HELLO (special command used for log-on)
LENGTH
TAPE
KEY

The following features are different in the two systems:

Statement	In BASIC/RT-11 V01
CALL	Implied call format (without the word CALL and the enclosing quotes) is not allowed.
IF THEN IF END#	The THEN or END# word cannot be followed by a statement; they can be followed only by a line number.
OPEN	No expressions are allowed; all numbers must be constants and all strings must be either string constants or scalar string variables. The MODE, FILESIZE, and RECORDSIZE options are not available. Filesize is specified by a number enclosed in quotes after FOR OUTPUT. The number sign (#) is mandatory. Virtual Files: subscript cannot be greater than 32,767; the maximum length of string virtual file elements can only be 1, 2, 4, 8, 16, 32, 64, or 128 characters. Nonfile-structured OPEN is not allowed; a default filename is assumed. The extension ".DAT" is always assumed unless an extension is specified.
OVERLAY	Line number can not be specified; program name is changed.
CLOSE RESTORE	No expressions are allowed; all numbers must be constants. The number sign (#) is mandatory.
REM	When in a multiple statement line, can only be the last statement.
PRINT# INPUT#	Only a colon (:) can be used after the expression. In MU BASIC/RT-11 either a colon or comma can be used.
Function	
CHR$	Outputs only a 7-bit value. In MU BASIC/RT-11 an 8-bit value is output.
Command	
OLD SAVE REPLACE	Require quotes around the file descriptor; these commands always assume the extension ".BAS" unless another is specified.
RUN	A file descriptor cannot be specified.

There are also the following differences:

BASIC/RT-11 V01	MU BASIC/RT-11
Maximum line number is 65532.	Maximum line number is 32767.
The ampersand (&) is the only operator signifying string concatenation.	Both the & and + signs indicate string concatenation.
Terminal width is always 72 characters.	Terminal width can be set in the range 1 to 255.
CTRL/C does not print the line number and returns control to the monitor.	CTRL/C prints the line number and returns to the READY message.
Files are closed by STOP and END statements and RUN, SCR, OLD, and NEW command.	Files are closed by the END statement (or execution of the last line of the program) but are left open by the STOP statement. Files are purged by the RUN, SCR, OLD, and NEW commands.
ALTMODE is equivalent to CTRL/U and echoes as "DELETED".	Only CTRL/U is valid command and it echoes as ↑U.
←is equivalent to RUBOUT.	Only RUBOUT is a valid command.

INDEX

HOW TO OBTAIN SOFTWARE INFORMATION

SOFTWARE NEWSLETTERS, MAILING LIST

The Software Communications Group, located at corporate headquarters in Maynard, publishes software newsletters for the various DIGITAL products. Newsletters are published monthly, and keep the user informed about customer software problems and solutions, new software products, documentation corrections, as well as programming notes and techniques.

There are two similar levels of service:

- The Software Dispatch
- The Digital Software News

The Software Dispatch is part of the Software Maintenance Service. This service applies to the following software products:

PDP-9/15
RSX-11D
DOS/BATCH
RSTS-E
DECsystem-10

A Digital Software News for the PDP-11 and a Digital Software News for the PDP-8/12 are available to any customer who has purchased PDP-11 or PDP-8/12 software.

A collection of existing problems and solutions for a given software system is published periodically. A customer receives this publication with his initial software kit with the delivery of his system. This collection would be either a Software Dispatch Review or Software Performance Summary depending on the system ordered.

A mailing list of users who receive software newsletters is also maintained by Software Communications. Users must sign-up for the newsletter they desire. This can be done by either completing the form supplied with the Review or Summary or by writing to:

Software Communications
P.O. Box F
Maynard, Massachusetts 01754

SOFTWARE PROBLEMS

Questions or problems relating to DIGITAL's software should be reported as follows:

North and South American Submitters:

Upon completion of Software Performance Report (SPR) form remove last copy and send remainder to:

Software Communications
P.O. Box F
Maynard, Massachusetts 01754

The acknowledgement copy will be returned along with a blank SPR form upon receipt. The acknowledgement will contain a DIGITAL assigned SPR number. The SPR number or the preprinted number should be referenced in any future correspondence. Additional SPR forms may be obtained from the above address.

All International Submitters:

Upon completion of the SPR form, reserve the last copy and send the remainder to the SPR Center in the nearest DIGITAL office. SPR forms are also available from our SPR Centers.

PROGRAMS AND MANUALS

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center.

Digital Equipment Corporation
Software Distribution Center
146 Main Street
Maynard, Massachusetts 01754

Digital Equipment Corporation
Software Distribution Center
1400 Terra Bella
Mountain View, California 94043

Outside of the United States, orders should be directed to the nearest Digital Field Sales Office or representative.

USERS SOCIETY

DECUS, Digital Equipment Computers Users Society, maintains a user exchange center for user-written programs and technical application information. The Library contains approximately 1,900 programs for all DIGITAL computer lines. Executive routines, editors, debuggers, special functions, games, maintenance and various other classes of programs are available.

DECUS Program Library Catalogs are routinely updated and contain lists and abstracts of all programs according to computer line:

- PDP-8, FOCAL-8, BASIC-8, PDP-12
- PDP-7/9, 9, 15
- PDP-11, RSTS-11
- PDP-6/10, 10

Forms and information on acquiring and submitting programs to the DECUS Library may be obtained from the DECUS office.

In addition to the catalogs, DECUS also publishes the following:

DECUSCOPE	-The Society's technical newsletter, published bi-monthly, aimed at facilitating the interchange of technical information among users of DIGITAL computers and at disseminating news items concerning the Society. Circulation reached 19,000 in May, 1974.
PROCEEDINGS OF THE DIGITAL EQUIPMENT USERS SOCIETY	-Contains technical papers presented at DECUS Symposia held twice a year in the United States, once a year in Europe, Australia, and Canada.
MINUTES OF THE DECsystem-10 SESSIONS	-A report of the DECsystem-10 sessions held at the two United States DECUS Symposia.
COPY-N-Mail	-A monthly mailed communique among DECsystem-10 users.
LUG/SIG	-Mailing of Local User Group (LUG) and Special Interest Group (SIG) communique, aimed at providing closer communication among users of a specific product or application.

Further information on the DECUS Library, publications, and other DECUS activities is available from the DECUS offices listed below:

DECUS
Digital Equipment Corporation
146 Main Street
Maynard, Massachusetts 01754

DECUS EUROPE
Digital Equipment Corp. International
(Europe)
P.O. Box 340
1211 Geneva 26
Switzerland

READER'S COMMENTS

NOTE: This form is for document comments only. Problems with software should be reported on a Software Problem Report (SPR) form (see the HOW TO OBTAIN SOFTWARE INFORMATION page).

Did you find errors in this manual? If so, specify by page.

_____
_____
_____
_____
_____
_____

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

_____
_____
_____
_____
_____
_____

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

_____
_____
_____
_____
_____
_____

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

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

If you do not require a written reply, please check here. ☐

Please cut along this line.

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

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

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

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

Postage will be paid by:

digital

Software Communications
P. O. Box F
Maynard, Massachusetts 01754

DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASSACHUSETTS 01754