DataGeneral

Customer Documentation

Business BASIC System Manager's Guide

Business BASIC
System Manager's Guide

093-000388-01

For the latest enhancements, cautions, documentation changes, and other information on this product, please see the Release Notice (085-series) supplied with the software.

Ordering No. 093-000388

Printed in the United States of America
Rev. 01, January 1989

Notice

Business BASIC System Manager's Guide
093-000388-01

Revision History:
Original Release - May, 1985
First Revision - January, 1989

Effective with:

AOS Business BASIC, Rev. 4.20
AOS/VS Business BASIC, Rev. 5.00
RDOS Business BASIC, Rev. 8.20
DG/RDOS Business BASIC, Rev. 8.20

A vertical bar in the margin of a page indicates substantive technical change from the previous revision.

Data General Corporation
4400 Computer Drive
Westboro, MA 01580

Preface

Scope

Data General's Business BASIC is a powerful, interactive programming language that runs on the following operating systems: mapped ECLIPSE® RDOS, DG/RDOS, AOS, AOS/VS, AOS/VS II, and AOS/WS. Unless a specific section indicates otherwise, this guide uses "AOS" to refer to AOS, AOS/WS, AOS/VS, and AOS/VS II. When differences exist, "AOS/VS" and "32-bit only" are used to refer to AOS/VS and AOS/VS II (the 32-bit operating systems), and "AOS" and "16-bit only" are used to refer to AOS and AOS/WS (the 16-bit operating systems). Similarly, "RDOS" refers to RDOS and DG/RDOS unless otherwise noted.

This manual describes the software delivered in a Business BASIC package and how to prepare for its use. The manual describes how to load, generate, and operate AOS and RDOS Business BASIC systems. It presumes you are a system manager or system operator who is familiar with Business BASIC and your operating system. If you need additional information on the Business BASIC language or on the operating systems, see "Related Documents" at the end of this manual.

Organization

This manual is divided into seven chapters and six appendixes. Each chapter is divided into an AOS section and an RDOS section. Some appendixes also have separate sections for AOS and RDOS.

Chapter 1 introduces Business BASIC.

Chapter 2 describes the contents of the Business BASIC package.

Chapter 3 describes how to generate Business BASIC.

Chapter 4 describes how to execute Business BASIC.

Chapter 5 describes the security features for Business BASIC.

Chapter 6 describes the Business BASIC utilities.

Chapter 7 lists the privileged system calls (STMBs, STMCs, STMDs, and STMEs), which can be used as statements and commands. Only system managers who know the password can enter these system calls into a program.

Appendix A contains information about the user status tables for AOS and RDOS. You usually use these tables with the STMB statements and commands described in Chapter 7.

Appendix B contains the ASCII character set.

Appendix C contains the Function Key Character Set.

Appendix D lists Business BASIC and operating system error messages.

Appendix E is a quick reference table of privileged system calls.

Appendix F describes the differences among terminal types in Business BASIC.

Syntax Conventions

The conventions used in this manual are described below.

UPPERCASE	Indicates a Business BASIC command, statement, or function.
lowercase	Indicates a generic term representing a complete syntactical entry to be supplied by the programmer.
Hyphen (-)	Between words, indicates a complete entry to be supplied by the programmer.
{ }	You must select one of the arguments in braces; do not enter the braces themselves.
[ ]	Optional arguments are set in brackets; do not enter the brackets.
...	Indicates that the preceding item can be repeated.

End of Preface

Contents

Chapter 1 — Introduction

Chapter 2 — The Business BASIC Package

Chapter 3 — Business BASIC System Generation

Chapter 4 — Executing Business BASIC

Chapter 5 — Security

Chapter 6 — Utilities

Chapter 7 — Privileged System Calls

Appendix A — User Status Tables

Appendix B — ASCII Character Set

Appendix C — Function Key Character Set

Appendix D — Error Messages

Appendix E — Summary Tables of Privileged System Calls

Appendix F — Terminal Types

Related Documents

Tables

Table

Figures

Figure

Chapter 1
Introduction

This chapter describes some features of AOS and RDOS Business BASIC.

AOS is a multiprogramming system in which many programs can run concurrently. Each executing program along with its set of system resources is a process. Each user's process resembles a complete computer system: it often has a distinct programming console; it can use all system devices; and, in some cases, it can create many son processes that have their own resources.

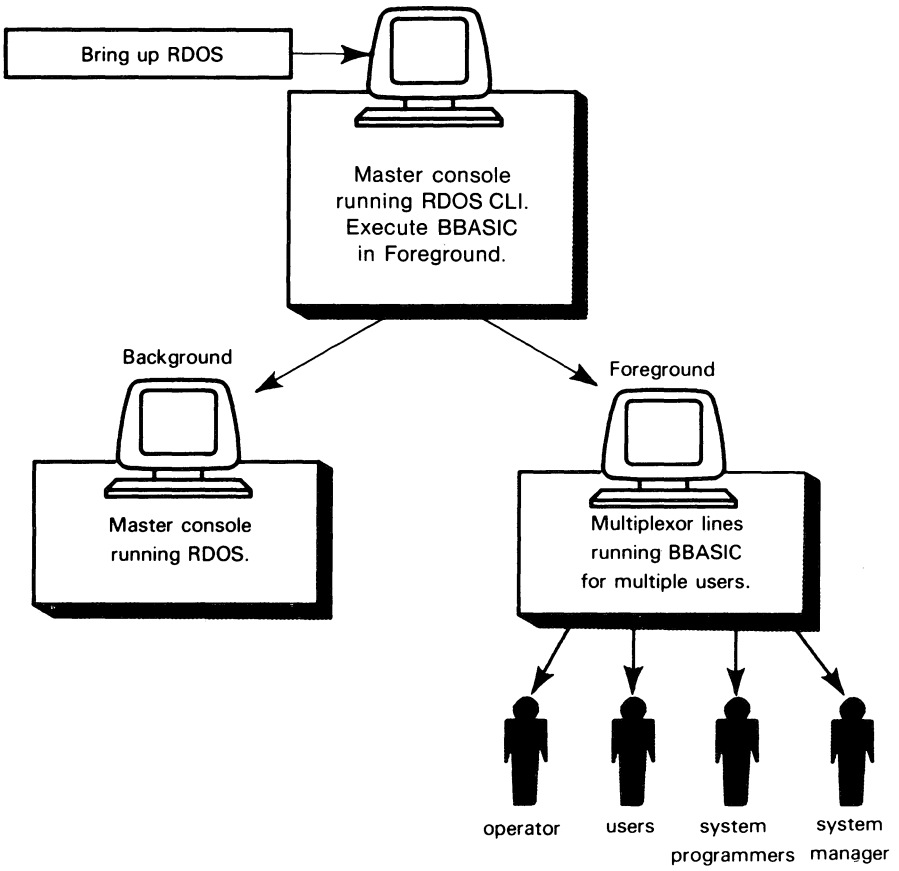
Under AOS, processes are independent of each other; individual processes usually cannot affect other processes. Each process offers multitasking, so that each user can perform several different tasks concurrently, and each task can respond individually to its own environment. Different processes can execute different versions of Business BASIC (including a run-only version) as well as other AOS utility programs.

Business BASIC security under AOS is handled primarily by the operating system. The operating system maintains a username/password security scheme, where any son process you create bears your username. Business BASIC maintains a User Status Table associated with your username.

Under RDOS, Business BASIC can run in either the foreground or the background and handle multitasking to support multiple users. Since each system is unique, you must tailor RDOS for your own environment. Before generating RDOS, become familiar with Business BASIC's requirements. For example, a Business BASIC system on RDOS has special requirements for stacks, cells, and buffers. Business BASIC can also handle multiplexors for multiple users. You can generate this support in Business BASIC or in RDOS.

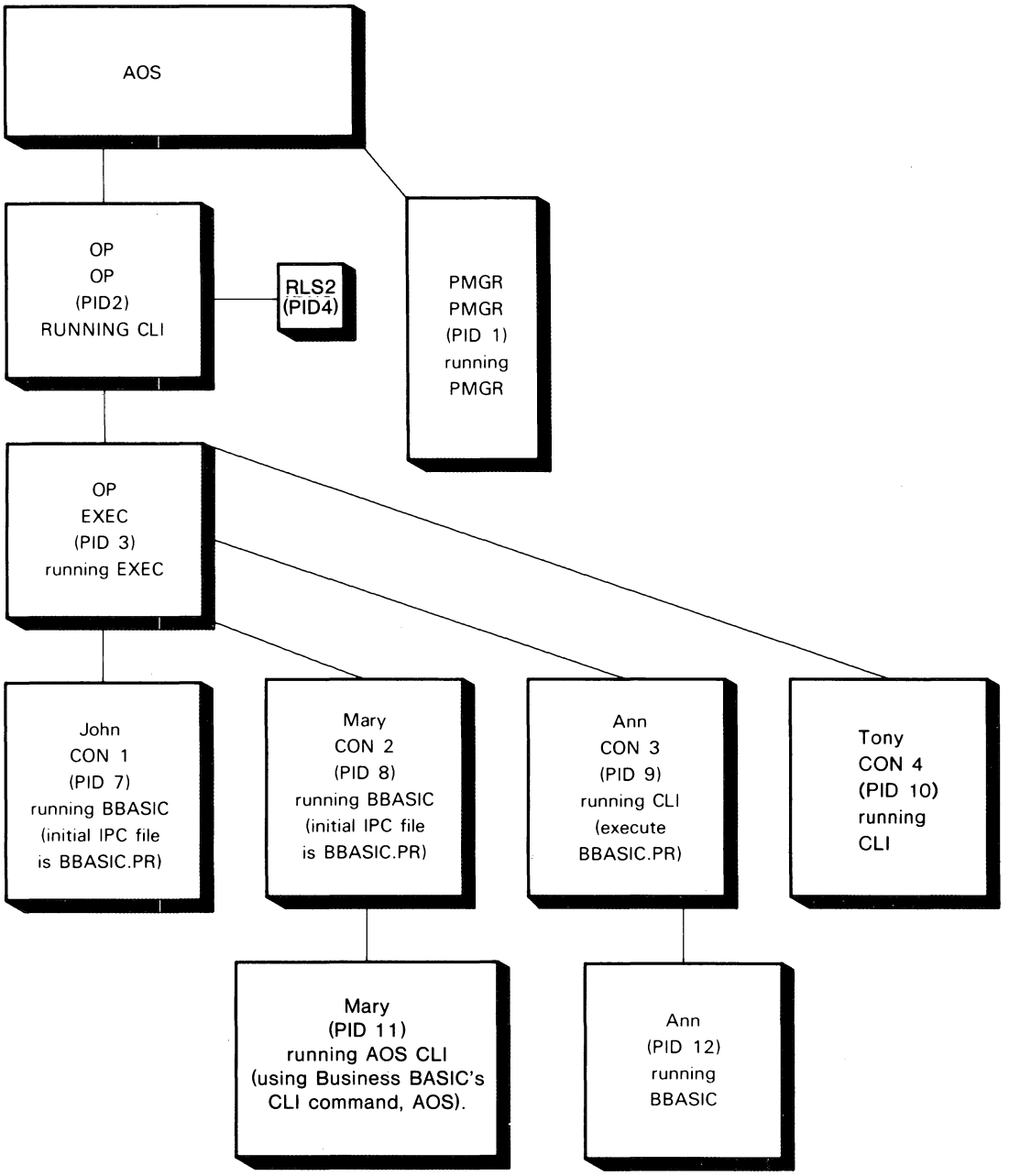
Business BASIC becomes a subsystem running in an RDOS environment. It has its own CLI to handle access to devices, files, and RDOS CLI operations, as well as an OPCLI program to handle operator responsibilities.

See Figure 1-1 for a simplified diagram of an RDOS Business BASIC system and Figure 1-2 for a simplified diagram of AOS Business BASIC processes. Refer to your operating system's manual for explanations and generating procedures.



UC-2000

Figure 1-1 RDOS Business BASIC



UC-2001

Figure 1-2 AOS Business BASIC

End of Chapter

Chapter 2
The Business BASIC Package

This chapter describes the files and directories delivered in your Business BASIC software package, and explains how you can manage those files.

The AOS Business BASIC Package

AOS Business BASIC is available on various media. See your release notice for information about the medium used with your operating system and hardware.

Contents

The AOS Business BASIC package contains the following files and directories.

File/Directory	Contents
Release notice	A description of the changes in the current release and information on installing the release. The name of this file is in the form 085_nnnnnn_nn.
BBASIC.OL and BBASIC.PR	The default double precision system. (BBASIC.OL is not used for AOS/VS or AOS/VS II.)
RLS2.PR and RLS2.ST	The resource lock server.
$DOC	Contains source listings of some of the utility programs and terminal modules, documentation update files, and demonstration programs for INFOS® II.
$SYSLIB	Double precision Business BASIC system utilities. (Required to execute double precision Business BASIC.)
$SYSLIB3	Triple precision Business BASIC system utilities. (Required to execute triple precision Business BASIC.)

File/Directory	Contents
NBASIC	Double precision demonstration files.
NBASIC3	Triple precision demonstration files.
BASICGEN	Files to generate a new double precision Business BASIC system. (Required to generate double precision Business BASIC.)
BASICGEN3	Files to generate a new triple precision Business BASIC system. (Required to generate triple precision Business BASIC.)
$MISC	Miscellaneous unsupported Business BASIC programs.

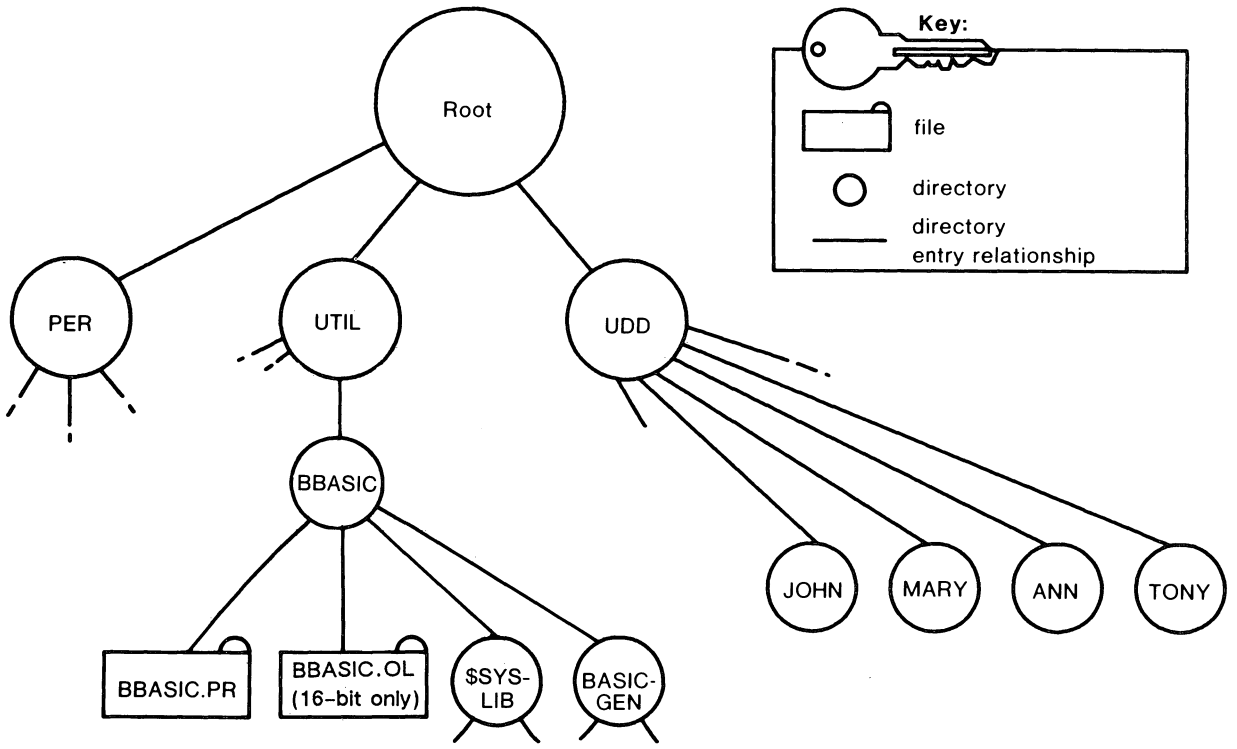
You may wish to delete any unused directories to save disk space.

Loading the Business BASIC Media

For detailed information on loading Business BASIC from magnetic tape or other media, refer to your release notice.

For convenience, the examples in this book assume that you load Business BASIC software into the directory :UTIL:BBASIC. However, this is not required; you may load the software into some other directory if you prefer.

If you plan to load this new release into a directory that contains an earlier version of Business BASIC, you should first back up the earlier version, and then delete it from the directory before loading the new revision. You can thus prevent the mixing of utilities from the earlier revision with those of this revision.



UC-2002

Figure 2-1 A Sample AOS Directory Tree

Resource Lock Server

Record locking for Business BASIC processes is coordinated through the use of a global Resource Lock Server program (RLS2). RLS2 keeps lock information in a database that can be accessed by Business BASIC processes as well as RLS2. This allows Business BASIC processes to enter a lock request in the RLS2 shared database locally; if this is successful, no interprocess communications (IPCs) to RLS2 are necessary, and IPC overhead is avoided. If the local lock request is unsuccessful, IPCs are used to ask RLS2 to enter the lock request.

Business BASIC processes try to cut down on IPC traffic between RLS2 and other Business BASIC processes. For example, when a Business BASIC process unlocks a record that another process is waiting for, the unlocking process sends an IPC directly to the waiting process. The waiting process and the unlocking process must have IPC privileges for this shortcut to be activated. For this reason, all Business BASIC users should have the IPC privilege.

When invoking Business BASIC, users may elect to use the /N switch. This switch tells Business BASIC not to allocate a shared page to allow local locking of the shared database; instead locking is handled exclusively by RLS2, and all communication with RLS2 is through IPCs. Business BASIC gains 2 Kbytes of

program space when the /N switch is used, but more CPU overhead is associated with this method. In low activity locking environments or in INFOS II-only systems, the program space gain may outweigh the performance degradation.

It is not necessary to create the RLS2 process in any particular directory, nor is it necessary to have SUPERUSER privileges to create the RLS2 process. However, you must allow all users to have read access to the directory where you created RLS2. Also, since there is no restriction on where RLS2 is created, there may be more than one RLS2 process running on a single system. One RLS2 may be accessed by test programs under development, for example, while another is accessed by live application programs. If you have more than one RLS2 process running, make sure that the search list is set up correctly so that each program accesses the desired RLS2. The HELLO program displays the pathname of the RLS2 IPC file that is being used, or the message `RLS2 NOT FOUND` if no RLS2 IPC file was found in either the working directory or a search list directory.

RLS2 creates three files during initialization:

- RLS2_REQUEST is an IPC entry for LOCK/UNLOCK processing.
- RLS2 is an IPC entry for operator commands (such as STOP).
- RLS2.COMM is the shared database for lock and wait entries.

To use record locking, you must be running the RLS2 process. The following macro is an example of how to initialize the RLS2 process. You can add these lines to your system's UP macro.

```
PUSH
WRITE INITIATING RLS2 LOCK SERVER PROCESS
SUPERUSER ON
DELETE/V/2=IGNORE :UTIL:BBASIC:RLS2<,.COMM,_REQUEST>
SUPERUSER OFF
PROC/DEFAULT/NAME=RLS2/CALLS=3/RESIDENT/PRIORITY=2 :UTIL:BBASIC:RLS2.PR
POP
```

NOTE: The maximum number of concurrent locks is 584 for AOS/VS Business BASIC and 85 for AOS Business BASIC. This cannot be altered.

Before bringing your system down, you must end the RLS2 process so that it closes its files. You can either issue the following command from your operator console or add it to your system's DOWN macro:

```
CONTROL RLS2 STOP
```

When the command is processed, RLS2 closes its files and terminates. This command works only if the username of the process issuing the command matches the username of the process that created RLS2.

Do not use the operator command to terminate a Business BASIC process that interfaces with RLS2 if the Business BASIC process is modifying the RLS2 database

or updating an index file. In the first case, using the operator command could cause all Business BASIC processes issuing locks to hang. In the second case, it could corrupt the index file. In addition, users should avoid the indiscriminate use of ^C^B or ^C^E to exit Business BASIC when record locking is being employed.

In addition, RLS2 checks only the active lock list for conflicts instead of checking both the wait list and the lock list. Because locking under AOS Business BASIC works the same way as it does under RDOS Business BASIC, you can avoid the following "deadly embrace" situation:

1) Process A issues LOCK 1,"file",0,512 -> A locks area

2) Process B issues LOCK 1,"file",0,512 -> B waits on A

3) Process A issues LOCK 2,"file",0,512 -> A waits on B

With use of the resource lock server, A's second LOCK request to the same area (in step 3 above) is accepted and does not wait on B.

If you are running AOS/WS, the macro DOWN.BBASIC.CLI is provided.

Loading a Business BASIC Update

An update does not usually contain the entire product; it contains updated versions of files found on the previous full release and perhaps some entirely new files. Installing an update is the same as installing a full release, with a few exceptions. Refer to your update notice for detailed information on loading a specific Business BASIC update. The following general guidelines apply to all Business BASIC updates:

- Do not delete the current Business BASIC directories before loading the update.

- Any files that you have modified and that are present on the update must be re-modified.

The RDOS Business BASIC Package

RDOS Business BASIC is supplied on various media. Refer to your release notice for the medium used with your operating system and hardware.

Contents

Your Business BASIC package contains the following directories and files.

Directory or File	Contents
Release notice	A description of the changes in the current release and information on installing the release. The name of this file is in the form 085nnnnnn.nn.
$SYS	Business BASIC system utilities for a double precision system. (Required to execute double precision Business BASIC.)
$LIB	Business BASIC utility programs and subroutines for a double precision system. (Required to execute double precision Business BASIC.)
$SY3	Business BASIC system utilities for a triple precision system. (Required to execute triple precision Business BASIC.)
$LIB3	Business BASIC utility programs and subroutines for a triple precision system. (Required to execute triple precision Business BASIC.)
BASICGEN	All files necessary to generate a double precision Business BASIC system. (Required to generate double precision Business BASIC.)
BASICGEN3	All files necessary to generate a triple precision Business BASIC system. (Required to generate triple precision Business BASIC.)
$SPL	Spooler for a double precision system. (Required for the double precision Business BASIC spooler.)
$SPL3	Spooler for a triple precision system. (Required for the triple precision Business BASIC spooler.)
NBASIC	Double precision demonstration files.

Directory or File	Contents
NBASIC3	Triple precision demonstration files.
$DOC	Source listings for utility programs and terminal modules, and documentation update files.
$MISC	Miscellaneous unsupported Business BASIC programs.

You may wish to delete any unused directories to save disk space.

Loading the Business BASIC Media

For detailed information on loading from magnetic tape or other media, refer to your release notice. For information about device mnemonics, see your operating system documentation.

Your current directory should be the primary or secondary partition where you want the Business BASIC system to reside. If you plan to load this new release into a directory that contains an earlier version of Business BASIC, you should first back up the earlier version, and then delete it from the directory before loading the new revision. You can thus prevent the mixing of utilities from the earlier revision with those of this revision.

Loading a Business BASIC Update

An update to Business BASIC usually does not contain the entire product; it contains updated versions of files found on the previous full release and perhaps some new files. You install an update the same way you install a full release, with a few exceptions. Refer to your update notice for detailed information on loading a specific Business BASIC update. The following general guidelines apply to all Business BASIC updates:

- Do not delete the current Business BASIC directories before loading the update.
- Any files that you have modified and that are present on the update must be re-modified.

End of Chapter

Chapter 3
Business BASIC System Generation

When you generate a Business BASIC system, you define the features you want to use. After generating the system, you have a Business BASIC system that can be executed.

How to Generate AOS Business BASIC

After meeting the following requirements for generating Business BASIC, answer the system generation prompts. Study the procedure and select the options you want before you begin.

Requirements

Before running Business BASIC, generate your AOS operating system using the instructions provided with AOS and then perform the following operations:

1) Become the user OP or turn SUPERUSER ON.

2) Make the AOS utility LINK.PR available by adding :UTIL to your search list.

Procedure

To generate AOS Business BASIC, start at the AOS CLI, use DIR to move to the BASICGEN directory for double precision or to BASICGEN3 for triple precision, and use the BBASIC_BUILD.CLI macro to create your Business BASIC system. See below.

```
) DIR :UTIL:BBASIC:BASICGEN
) BBASIC_BUILD
```

The BBASIC_BUILD macro creates an individual Business BASIC system for you by prompting you to choose options for several Business BASIC characteristics. This macro executes the BASIC System Generation (BSG) program to produce a BUILD_systemname.CLI macro that links the appropriate modules for your system. Before executing the BSG program, the BBASIC_BUILD macro checks for the link

program and the system generation files: LINK.PR, BSG.Q, and BSG.L. If any of these files is missing, the macro displays an error message and terminates.

If all the required files are found, the macro then prompts you to choose from options for several Business BASIC characteristics. Press the New Line key to accept the default response shown in brackets. Enter Y or N (in uppercase) to select an option.

The macro begins by displaying:

```
Business BASIC sysgen program: Rev X.XX

 Enter system name:
```

Specify the name you want assigned to your Business BASIC system.

When a Business BASIC system is generated with the same name as an existing Business BASIC system, some existing files that correspond to that system name are deleted and new files are created. They are the systemname.PR, systemname.OL (16-bit only), systemname.MP and systemname.ST files. The existing BUILD_systemname.CLI and systemname.SG files are modified.

The dialog for BBASIC_BUILD.CLI is as follows:

```
Dialog for AOS Business BASIC Revision X.XX

1. Do you want a run-only system?
    [N] (N,Y):
```

A Business BASIC run-only system allows only program execution. The prompt * RUN indicates a run-only system. (An asterisk (*) prompt indicates a regular Business BASIC system.) Run-only system users cannot ENTER or ERASE statements or LIST programs.

```
2. Do you want to include the INFOS II statements?
    [N] (N,Y):
```

Answer Y to allow the use of the Business BASIC INFOS® interface. Answering N allows a larger program size in 16-bit Business BASIC and prohibits the use of the Business BASIC INFOS "DB" statements (DBOPEN, DBCLOSE, etc.).

To include the INFOS II statements, use the current version of ICALL.OB (or ICALL32.OB on AOS/VS) that is provided with INFOS II. This module or a link to it must be in BASICGEN (or BASICGEN3) or must be in a directory on your search list at the time your Business BASIC system is linked.

The directory $DOC contains the demonstration programs BBII_DEMO1.BA, BBII_DEMO2.BA, and BBII_DEMO3.BA, which illustrate the use of some Business BASIC INFOS II statements.

```
3. Do you want to include terminal types 8 and 9?
   [N] (N,Y):
```

Type 8 has a default page width of 80 characters. Type 9 has a default page width of 132 characters. Both use the AOS screen management primitives. To use both terminal types, respond Y, and execute Business BASIC with the global /E switch. The default is terminal type 6, which has a default page width of 80 characters but does not use AOS screen management primitives.

```
4. Do you want all users to be able to use privileged statements?
   [N] (N,Y):
```

Respond Y to allow all users to enter STMBs, STMCs, and STMEs. Business BASIC then disables checking for a privileged (AA) account.

If you are generating the AOS/VS version of Business BASIC, you must answer the following system generation question. If you are generating an AOS or AOS/WS version of Business BASIC, this question is **not** part of your system generation process.

```
5. Do you want to include the debugging statements?
 [N] (N,Y):
```

Answer Y to allow the use of Business BASIC debugging features, such as TRACE and STEP. You should respond Y when you need a development environment. Choosing this option may have some effect on speed. Answering N prohibits the use of the Business BASIC debugging statements and commands. Use N for security reasons.

After you have completed the system generation dialog, a BUILD_systemname.CLI macro and a systemname.SG dialog file are produced. The BBASIC_BUILD macro then prompts:

```
Do you want to link the system now?
 [Y] (Y,N):
```

Respond Y to create an executable Business BASIC system. Respond N to save this file for later editing or linking.

```
Do you want to save these command lines?
 [Y] (Y,N):
```

Answer Y to leave your command lines and their responses in BUILD_systemname.CLI. You may relink this Business BASIC system by running this macro. Respond N to delete BUILD_systemname.CLI.

```
Your sysgen dialog is in file systemname.SG.
```

After building your Business BASIC system, move the systemname.PR and systemname.OL (16-bit only) files to the :UTIL:BBASIC directory before executing Business BASIC.

Including Assembly Language Subroutines in AOS and AOS/WS (16-Bit) Business BASIC

You can increase Business BASIC functions with assembly language subroutines. To use them, you generate a version of Business BASIC that includes an interface for the subroutines. Users then access these subroutines with the UCALL statement or command.

The steps you follow to implement assembly language subroutines are:

1) Put the assembly subroutines in USERSUBS.SR.

2) Use BBUMASM.CLI to produce USERSUBS.OB.

3) Use BBASIC_BUILD.CLI to produce a new system.

The module USERSUBS is automatically loaded into any Business BASIC interpreter. The default USERSUBS module supplied on your release media defines no routines. To implement a UCALL, you should edit and assemble the file USERSUBS.SR in the BASICGEN (or BASICGEN3) directory, and then generate or relink your system. Whenever a system is built or relinked, the USERSUBS.OB module in the BASICGEN directory is included.

The USERSUBS.SR module on the release media contains a commented-out sample user subroutine. This subroutine employs several macros that simplify the writing of user subroutines. The macros are defined in parameter files supplied with the release media.

A CLI macro named BBUMASM.CLI is provided with the release media to simplify the assembly of user-written subroutines. The BBUMASM macro builds the necessary permanent symbol table using parameter files in the BASICGEN (or BASICGEN3) directory. BBUMASM also uses operating system parameter files and utilities, which must be in a directory on your search list. (They are normally in :UTIL.)

At the AOS CLI prompt, assemble USERSUBS.SR by entering:

```
) SUPERUSER ON
*) SEARCHLIST :UTIL : [!SEARCHLIST]
*) DIR :UTIL:BBASIC:BASICGEN
*) BBUMASM USERSUBS
```

When you execute the BBUMASM USERSUBS command, Business BASIC creates

the permanent symbol table called BBU_MASM.PS if it does not already exist. Using a /PS switch on BBUMASM forces the deletion of the existing symbol table and the creation of a new symbol table. You can find a listing of the assembled module in USERSUBS.LS.

Numeric and string variables and expressions can be the arguments for a UCALL; however, only single elements of numeric arrays can be used. All variables must be dimensioned (DIM) or initialized before the UCALL.

A UCALL can have up to eight arguments. The start of a UCALL and its arguments are defined by the ASMDEF macro. The form is:

ASMDEF number,argument-1,...,argument-n

where:

number — Is the subroutine number. For example, 10. is the number for UCALL 10—the default radix for MASM is octal; thus the decimal point indicates a decimal number.

argument-n — Is the type of the nth argument. Possible values are:

SEX	String expression
NEX	Numeric expression
SV	String variable
NV	Numeric variable

The location immediately after the ASMDEF statement should be the start of the subroutine.

After defining the UCALL with ASMDEF, use the DEFDMP macro to create the subroutine table. The interpreter refers to this table when evaluating arguments in preparation for the call. The table must be at location SBRTB. Thus, end your USERSUBS module with:

```
SBRTB: DEFDMP
```

Business BASIC passes two pointers to the subroutine. The first is a word pointer to an argument list; this pointer is in accumulator 2 and also in the ZREL location TR1. Note that the pseudo-op .EXTD must be coded if you refer to TR1 or any other special ZREL locations. The second pointer is the return address that is used at the end of your UCALL. To return, either JMP @TR15 (and code .EXTD TR15) or JMP to the word immediately following the argument list.

The argument list contains a one-word entry for each argument. For numeric arguments, the argument list entry is a word pointer to the high-order 16 bits of the value of the numeric variable or expression. Numeric values are stored internally in the following manner:

Double Precision	Triple Precision
High order	High order
Low order	Middle order
	Low order

For string expressions, the argument list entry is a word pointer to a two-word descriptor string that contains the following:

- Byte address of the first byte of the string
- Byte length of the string

For string variables, the argument list entry is a word pointer to a four-word descriptor string that contains the following:

- Byte address of the first byte of the string or substring
- Byte length of the string or substring
- Maximum byte length of the string or substring
- Word address of the current length of the string

For example, if A$ is dimensioned to 10 bytes and is completely full, and the argument to a UCALL is A$[6,10], the second and third words of the descriptor string both contain the value 5 and the fourth word is a pointer to a location that contains the value 10.

Business BASIC statements like PRINT and WRITE cannot use information beyond the current length of a string. If a result is to be returned in a string variable supplied in the UCALL, then the string variable should be null-filled before the UCALL is issued, so that the string variable is allocated. The UCALL should update the current length of the string after the information has been stored in the string.

For UCALL temporary storage areas, use:

- ZREL locations TR0 through TR15. They must be declared with .EXTD. Remember that TR1 and TR15 contain special pointers.
- ECLIPSE® stack instructions. If you change the stack, you must restore the original order of items in the stack. Use the EJSR RESBB instruction to reset the stack to its initial state.
- Local storage locations (if the code need not be re-entrant; for example, unshared NREL code).

To load constants into accumulators, use one of the following:

- The LEF or ELEF instruction
- The literal facility of the assembler
- Locally hard-coded constants

You can use USERSUBS as resident code, or you can load USERSUBS in an overlay by editing the LINK line. You can specify shared and unshared NREL code by using the .NREL pseudo-op or by the SC and UC local switches in the LINK line. When you use an overlay, the overlay must not cross a 4-Kbyte boundary.

Test each UCALL before it is linked into a production system. Because of the level on which a UCALL works, a UCALL with errors has great potential for causing system crashes and other serious faults. A UCALL is not always transportable from revision to revision or from operating system to operating system.

Including Assembly Language Subroutines in AOS/VS and AOS/VS II (32-Bit) Business BASIC

With 32-bit Business BASIC, you must assemble user-written assembly language subroutines using the 32-bit macroassembler (MASM).

When coding assembly language subroutines, remember that packets often vary between 16-bit and 32-bit calls. For more information, refer to the appropriate AOS/VS manuals:

- *AOS/VS System Concepts*
- *System Call Dictionary (AOS/VS and AOS/DVS)*
- *AOS/VS Macroassembler (MASM) Reference Manual*
- *Eclipse 32-Bit Systems Principles of Operation Programmer's Reference*

The steps you take to implement assembly language subroutines for 32-bit Business BASIC are the same as for 16-bit Business BASIC:

- Place the assembly subroutines in USERSUBS.SR.
- Use BBUMASM.CLI to produce USERSUBS.OB.
- Use BBASIC_BUILD.CLI to produce a new system.

The module USERSUBS is automatically loaded into any Business BASIC interpreter. The default USERSUBS module supplied on your release media defines no routines. To implement UCALLs, you should edit and assemble the file USERSUBS.SR in the BASICGEN (or BASICGEN3) directory, and then generate or relink your system. Whenever a system is built or relinked, the USERSUBS.OB module in the BASICGEN directory is included.

The USERSUBS.SR module on the release media contains two commented-out sample user subroutines. These subroutines employ several macros that simplify the writing of user subroutines. The macros are defined in parameter files supplied with the release media.

A CLI macro named BBUMASM.CLI is provided with the release media to simplify the assembly of user-written subroutines. The BBUMASM macro builds the necessary permanent symbol table using parameter files in the BASICGEN (or BASICGEN3) directory. BBUMASM also uses operating system parameter files and utilities, which must be in a directory on your search list. (They are normally in :UTIL.)

At the AOS CLI prompt, assemble USERSUBS.SR by entering:

```
) SUPERUSER ON
*) SEARCHLIST :UTIL : [!SEARCHLIST]
*) DIR :UTIL:BBASIC:BASICGEN
*) BBUMASM USERSUBS
```

When you execute the command BBUMASM USERSUBS, Business BASIC creates the permanent symbol table called BBU_MASM.PS if it does not already exist. Using a /PS switch on BBUMASM forces the deletion of the existing symbol table and the creation of a new symbol table. You can find a listing of the assembled module in USERSUBS.LS.

Numeric and string variables and expressions can be the arguments for a UCALL; however, only single elements of numeric arrays can be used. All variables must be dimensioned or initialized before the UCALL.

A UCALL can have up to eight arguments. As with 16-bit Business BASIC, the ASMDEF and DEFDMP macros are used to describe the UCALL arguments. However, use of the 32-bit ASMDEF and DEFDMP macros does differ from use of the 16-bit macros. For example, since 32-bit MASM has a SEX instruction, the mnemonic for "string expression" is STRGEX.

The start of a UCALL and its arguments are defined by the ASMDEF macro. The form is:

ASMDEF number,argument-1,...,argument-n

where:

number — Is the subroutine number. For example, 10. is the number for UCALL 10—the default radix for MASM is octal; thus the decimal point indicates a decimal number.

argument-n — Is the type of the nth argument. Possible values are:

- STRGEX — String expression
- NEX — Numeric expression

SV	String variable
NV	Numeric variable

The location immediately after the ASMDEF statement should be the start of the subroutine.

After defining the UCALL with ASMDEF, use the DEFDMP macro to create the subroutine table. The interpreter refers to this table when evaluating arguments in preparation for the call. The table must be at location SBRTB. Thus, end your USERSUBS module with:

```
SBRTB: DEFDMP
```

In the four-word table produced for each UCALL by DEFDMP, word 1 is zeroed and is reserved. The structure of the 4-word table is:

Table 3-1 Four-Word UCALL Table Produced by DEFDMP

Word	Value
0	UCALL #
1	0 (reserved)
2	Max args | Min args
3	Variable control word

In accumulator 2 (ac2), Business BASIC passes a word pointer to an argument list; this pointer is also in location UCALL_POINTER. Location UCALL_ARG_CNT is a one-word location containing the number of arguments passed to the UCALL.

To return to the interpreter, execute the command LJMP UCALL_RETURN. You cannot return by jumping to the word following the argument list.

Your UCALL routine can return an error code. Do this by loading the appropriate error code from PS_PARBU.SR into ac2 and then executing the command LJMP UCALL_ERROR. See the routine ERROR_EXIT in USERSUBS.SR for an example of how to code an error return routine.

The argument list passed to a UCALL is made up of double words. For numeric arguments, the entry is a word pointer to the 16-bit word containing the high-order value of the numeric variable or expression. With regard to precision, the internal storage of the values is as follows:

Double Precision	Triple Precision
High order	High order
Low order	Middle order
	Low order

For string expressions, the argument list entry is a word pointer to a descriptor string of two double words. The first double word contains the byte address of the first byte of the string; The second contains the byte length of the string.

For string variables, the argument list entry is a word pointer to a descriptor of four double words. It contains:

- Byte address of the first byte of the string or substring
- Byte length of the string or substring
- Maximum byte length of the string or substring
- Word address of the current length of the string

32-bit Business BASIC makes extensive use of the stack. Your subroutine must take care to preserve the frame pointer; if it is modified during execution of the subroutine, it must be restored prior to returning to the interpreter. As long as the frame pointer is preserved or restored, the subroutine can use the stack freely. Note that the RESSB routine, which is used to reset the stack in the 16-bit implementation, is not used in the 32-bit implementation.

You should place temporaries on the stack or in local storage variables. ZREL locations TR0 - TR15 do not exist in 32-bit Business BASIC. If you must use local storage for temporaries, start the symbols with the characters USR. to distinguish the temporaries from any symbols used in the Business BASIC interpreter. Place the temporaries in NREL unshared code partition 0 or 4 or in NREL unshared data partition 6.

Load constants using either the NLDAI or WLDAI instructions, or define locally hard-coded constants.

Sample System Generation Dialogs

Below is a sample AOS Business BASIC system generation for a 16-bit run-only system that includes an interface for INFOS II statements, includes terminal types 8 and 9, and allows all users to use privileged statements.

```
Business BASIC sysgen program: Rev X.XX

 Enter system name: BBINFOS

 Dialog for AOS Business BASIC Revision X.XX


 1. Do you want a run-only system?
    [N]  (N,Y): Y

 2. Do you want to include the INFOS II statements?
    [N]  (N,Y): Y

 3. Do you want to include terminal types 8 and 9?
    [N]  (N,Y): Y

 4. Do you want all users to be able to use privileged statements?
    [N]  (N,Y): Y
```

Below is a sample AOS Business BASIC system generation for a run-only system that does not use terminal types 8 and 9 or allow all users to use privileged statements. This system is not run-only and does not include INFOS II statements.

```
Business BASIC sysgen program: Rev X.XX

 Enter system name: BBASIC

 Dialog for AOS Business BASIC Revision X.XX


 1. Do you want a run-only system?
    [N]  (N,Y): N

 2. Do you want to include the INFOS II statements?
    [N]  (N,Y): N

 3. Do you want to include terminal types 8 and 9?
    [N]  (N,Y): N

 4. Do you want all users to be able to use privileged statements?
    [N]  (N,Y): N
```


Below is a sample AOS/VS Business BASIC system generation that includes terminal types 8 and 9, allows all users to use privileged statements, and includes the debugging statements. This system is not run-only and does not include INFOS II statements.

```
Business BASIC sysgen program: Rev X.XX

 Enter system name: TERMTYPE8

 Dialog for AOS/VS Business BASIC Revision X.XX



1. Do you want a run-only system?
    [N]  (N,Y): N

2. Do you want to include the INFOS II statements?
    [N]  (N,Y): N

3. Do you want to include terminal types 8 and 9?
    [N]  (N,Y): Y

4. Do you want all users to be able to use privileged statements?
    [N]  (N,Y): Y

5. Do you want to include the debugging statements?
    [N]  (N,Y): Y
```

How to Generate RDOS Business BASIC

To generate your Business BASIC system, first generate an RDOS operating system meeting Business BASIC's requirements and then execute the BSG program.

Requirements

Business BASIC requires that you generate your RDOS operating system with the following considerations:

1) Generate RDOS with enough channels for file I/O. You need:

   - At least three channels to support Business BASIC.
   - One channel for each Business BASIC reserved file. (See the BSG questions on reserved files in the "Procedure" section of this chapter.)
   - As many channels as required for individual users to open files simultaneously in their programs (up to 16 channels per user).
   - One channel for each multiplexor line you plan to use, if you use operating system multiplexor support (see requirement 8 below).
   - One channel for QTY:64, if you use operating system multiplexor support.
   - One channel for each user program library.

2) Generate your RDOS system with at least three stacks. The more Business BASIC users you expect to have, the more stacks you need to generate in RDOS. Use the default number of cells and 20 extra buffers.

3) To allow access to $SYS and $LIB (or $SY3 and $LIB3), generate your operating system so that at least two subdirectories can be initialized at the same time. Four subdirectories are necessary for simultaneous access to $SYS, $LIB, BASICGEN, and $SPL. In addition, you may want to add an extra subdirectory for each user.

4) If you will use a tape drive, generate it into RDOS.

5) Use a real-time clock with a frequency of 10 hz.

6) If you will use a system printer, generate it into RDOS.

7) You can generate your secondary console with operating system or Business BASIC support. For more information, see the "Second TTY" question in the "Procedures" section below. If you answer Y (yes) to the "Second TTY" question, **do not** generate your secondary console into RDOS.

8) If your Business BASIC system uses multiplexor support, you can generate this support in RDOS or Business BASIC. A conflict occurs if you generate support for a multiplexor in RDOS and also choose Business BASIC support for the same multiplexor.

   If you choose Business BASIC support, Business BASIC uses its own multiplexor drivers. Business BASIC multiplexor support allows maximum flexibility in choices of interrupt characters and less system call overhead, but it limits the multiplexor to use by only Business BASIC, as well as limiting the types of machines on which Business BASIC can be run. Business BASIC support is available only for ALM, ULM, ASLM, and USAM multiplexors; if you have other multiplexors, you must use operating system multiplexor support.

   If you choose operating system support, you must decide whether to use Ctrl-C Ctrl-x interrupt characters (available only in DG/RDOS) or single interrupt characters (an option in DG/RDOS, and the only choice in RDOS). Operating system multiplexor support allows Business BASIC to run anywhere RDOS runs, but this choice imposes certain restrictions on interrupt characters, and it entails more system call overhead.

   If you want to share a multiplexor between Business BASIC and another application, or if you want to have a printer on a multiplexor line known by RDOS, you must use operating system support for the multiplexor. If you have two multiplexors on different device codes, you can generate one in RDOS and one in Business BASIC. This system generation allows you to use Business BASIC multiplexor support and still have access to a printer on an operating system multiplexor line.

   Your Business BASIC application's requirements for interrupt handling may be critical in the choice of multiplexor support. Interrupt characters are handled differently with operating system multiplexor support than they are with Business BASIC multiplexor support. The following discussion of how interrupts are handled uses the terms REAL-TIME interrupt and READ-ONLY interrupt. A real-time interrupt refers to a character which, when typed, must result in an action immediately, regardless of whether a read is posted. (A read is said to be posted when an INPUT or TINPUT statement is executed or when the asterisk prompt is displayed.) A read-only interrupt is a character which, when typed, results in an action ONLY when a read is posted.

   Consider this Business BASIC code, which illustrates the difference in processing between real-time and read-only interrupts:

```
0010 ON IKEY THEN GOTO 200
0020 FOR I=1 TO 20000
.
```

```
0030   LET X=I*2
0040 NEXT I
0050 INPUT A
0060 STOP
0200 PRINT "IKEY"
```

If a real-time interrupt character is typed while the FOR-NEXT loop is executing, the program immediately exits the loop and prints "IKEY." If a read-only interrupt character is typed during the loop, the loop continues until it finishes, the INPUT statement is executed (causing a "?" prompt to appear), and then the interrupt occurs (causing "IKEY" to be printed). Therefore, an interrupt character must be real-time if you want it to take effect instantly rather than at the next read.

Under Business BASIC multiplexor support, Business BASIC immediately receives each character typed on each line, and immediately decides how to react to it. Thus, every key which transmits data to the multiplexor can function as a real-time interrupt. Business BASIC takes advantage of this situation by allowing you to choose three real-time interrupts from all the available keys: a primary interrupt character, a secondary interrupt character, and a detach character. Since you can dynamically redefine these real-time interrupts on each line of the multiplexor by using STMA 4, different terminals can use different interrupt characters simultaneously. With Business BASIC multiplexor support, all interrupts are real-time; none are read-only.

Under operating system multiplexor support, RDOS (not Business BASIC) immediately receives each character typed on each line, and RDOS immediately decides whether a character is a real-time interrupt. Your RDOS system generation determines how RDOS makes that decision. Either it recognizes a Ctrl-C followed by a Ctrl-x as an interrupt sequence, or it recognizes either of two single characters chosen during RDOS generation.

When you use Ctrl-C Ctrl-x interrupt characters, each interrupt character must be a two-key sequence where the first key is a Ctrl-C and the second is in the range Ctrl-A to Ctrl-Z. Because of their special meanings, Ctrl-S and Ctrl-Q cannot be used as interrupt characters. You can choose three real-time (Ctrl-C Ctrl-x) interrupts from the valid range: a primary interrupt character, a secondary interrupt character, and a detach character. Because you can use STMA 4 to dynamically redefine these interrupts on each line of the multiplexor, different terminals can be using different interrupt characters simultaneously. When you use STMA 4 to redefine an interrupt, the new interrupt character must also be in the range Ctrl-A to Ctrl-Z (excluding Ctrl-S and Ctrl-Q) if you want it to be a real-time interrupt. If you redefine the interrupt character outside this range, RDOS does not interpret it as a real-time interrupt character, which makes it a read-only interrupt character.

Even though you choose operating system multiplexor support with Ctrl-C Ctrl-x interrupts, you can have Business BASIC application programs that use the escape (<ESC>) key as the primary interrupt key. To do this,

specify Ctrl-C Ctrl-A as one of your interrupts during Business BASIC generation. Business BASIC opens a line in such a way that when you strike the <ESC> key a Ctrl-C Ctrl-A sequence is transmitted, thereby generating an interrupt from one keystroke.

When you use single interrupt characters instead of Ctrl-C Ctrl-x interrupt characters, you must generate the interrupt characters into RDOS. You can choose only two interrupt characters from all the keys which transmit data to the operating system. Since these two characters are the only real-time interrupts available to Business BASIC under this option for operating system support, you are limited to two real-time interrupts. You cannot have a primary interrupt character, a secondary interrupt character, and a detach character which all act as real-time interrupts. You must decide which two capabilities should be real-time interrupts, and which one can be read-only.

With single interrupt characters, all Business BASIC jobs are restricted to the same two interrupt characters. Although STMA 4 may be used to dynamically redefine a Business BASIC interrupt character, the new interrupt character must be one of the two interrupts generated into RDOS if you want it to be a real-time interrupt; any other character is interpreted as a read-only interrupt. If your Business BASIC application does not use the detach feature or does not need more than one interrupt character, this limitation may not be a problem for you.

The following table shows some differences between Business BASIC and operating system multiplexor ("mux") support with and without Ctrl-C Ctrl-x interrupts.

	BBASIC mux support	Operating system mux support ^C^x	Operating system mux support No ^C^x
# of real-time interrupts available	3	3	2
<ESC> is one-character interrupt	Yes	Yes	Yes
One-character interrupt other than <ESC> is available	Yes	No	Yes
Specific interrupt characters are generated in RDOS, not BBASIC	No	No	Yes
Mux may be shared between Business BASIC and another application	No	Yes	Yes

For more information about defining interrupt keys with your operating system, see the SYSGEN section in either *How to Load and Generate RDOS* or *How to Generate and Run DG/RDOS.*

9) Generate one user-defined device for power-fail/automatic-restart, one for each multiplexor device code (only for Business BASIC multiplexor support), and two for a secondary console .IDEFed by Business BASIC.

10) Before generating your Business BASIC system, you must have in your master directory either the following files (that come with your operating system release) or links to them, and the directory that they are in must have been initialized:

MAC.SV, MACXR.SV
RLDR.SV, RLDR.OL
NBID.SR, OSID.SR, DGPARU.SR, NEID.SR

NOTE: The DGPARU.SR file is named PARU.SR on all RDOS systems. On an RDOS system, you must link DGPARU.SR to PARU.SR.

Procedure

To generate a new Business BASIC system, bring up a Business BASIC interpreter. (A minimal one named BB is supplied in the BASICGEN and BASICGEN3 directories on your release media.) Then execute the BSG program, which is also in the BASICGEN and BASICGEN3 directories.

Before generating your system, make sure both your user and system directories are BASICGEN or BASICGEN3. To do this, use one of the following procedures:

1) If your system directory is not BASICGEN or BASICGEN3, use the DIR statement or command to change both your system and user directories to BASICGEN or BASICGEN3, as follows:

```
* DIR "BASICGEN (or BASICGEN3)
```

2) Execute Business BASIC from the BASICGEN or BASICGEN3 directory. Then specify the BASICGEN or BASICGEN3 directory at the DIRECTORY prompt during logon, as in the following generation example.

For more information about user and system directories, see the RDOS section "Directory, File and Link Access" in Chapter 5, "Security."

From the RDOS CLI, enter the following command to generate your Business BASIC system:

```
BB

DGC BBASIC X.XX
```

Press the Escape key; then answer the following prompts:

```
ACCOUNT: AAAAA
PASSWORD: DE2LA6 (not displayed)
DIRECTORY: BASICGEN (or BASICGEN3)
```

Then execute the BSG program by entering the following command from the Business BASIC CLI:

```
* !BSG[/E][/N] [systemname] [audit-file/A]
```

Arguments

systemname — The name of the system you will generate.

You can supply a name for your Business BASIC system or accept the default name, BBASIC. If you specify a name, you cannot include a directory specifier. The system must be created in the BASICGEN (or BASICGEN3) directory; it can be linked to or moved later. The new system name and the old audit-file name cannot be the same.

audit-file/A — Use the audit file from a previous BSG. It can be used with or without a global /E switch. If you don't use the /E switch, the system will be built automatically using the audit file you specify here, without displaying the generation dialog. The audit file has a .SG extension, but it need not be used here. Audit files are not transferable from revision to revision; create new ones with each revision. Do not use an audit file to generate your system if you are changing the multiplexor type.

Global Switches

/E Takes you through the BSG dialog, using answers from the existing audit file as defaults. Requires that an audit file be specified with a local /A switch.

/N Do not automatically assemble or load the system. The output is left in the files CLI.CM, systemname.SG, systemname.SR, and systemname.CM. If you do not use the /N switch, BSG automatically returns to RDOS and assembles and loads the new system.

When you enter the BSG command, the BSG program prompts you with a subset of the questions given below to determine the parameters for your Business BASIC system. For on-line help on any of the questions, type a question mark (?) in response to the question. For most questions, a help message is displayed. Otherwise, the message "Sorry, no help is available for this question" is displayed. Your responses to particular BSG questions determine your dialog. To give the default response shown within square brackets, enter a carriage return. If you make a mistake while answering questions in the BSG program, you can discard the answers you have entered so far and begin the BSG program again by performing the following actions:

1) Press Escape to exit BSG and return to the Business BASIC prompt.

2) Restart the BSG program.

Some of the following questions are asked only if you give a specific answer to one or more of the questions before them. Figure 3-1 shows the dependencies between questions.

```
Console terminal (Y or N) [Y]?
```

Respond Y if Business BASIC will use the operating system's master or secondary console. This is $TTO/$TTI when Business BASIC runs in the background and $TTO1/$TTI1 when Business BASIC runs in the foreground. This is Business BASIC port 0.

```
Second TTY (Y or N) [N]?
```

Respond Y to this question if you will run this Business BASIC system in the background and would like to use the secondary console as a Business BASIC port from the background. This is Business BASIC port 1. Before you can answer Y, three conditions must be satisfied:

- Your system must have a secondary console on device codes 50 and 51.
- This secondary console must not be generated into RDOS.
- If a foreground program is running, it cannot use $TTO1/$TTI1.

If you answer Y to the Second TTY question, the following modem question is displayed:

```
Modem control (Y or N) [N]?
```

Respond Y if the secondary console has a modem control option and you want to use it.

```
Use operating system multiplexor support (Y or N) [N]?
```

Respond Y to this question to include the routines necessary to run with operating system multiplexor support. When you select this option, you must generate support for the multiplexors using the SYSGEN program of RDOS or DG/RDOS. To set characteristics for individual lines, follow the instructions in your operating system manual.

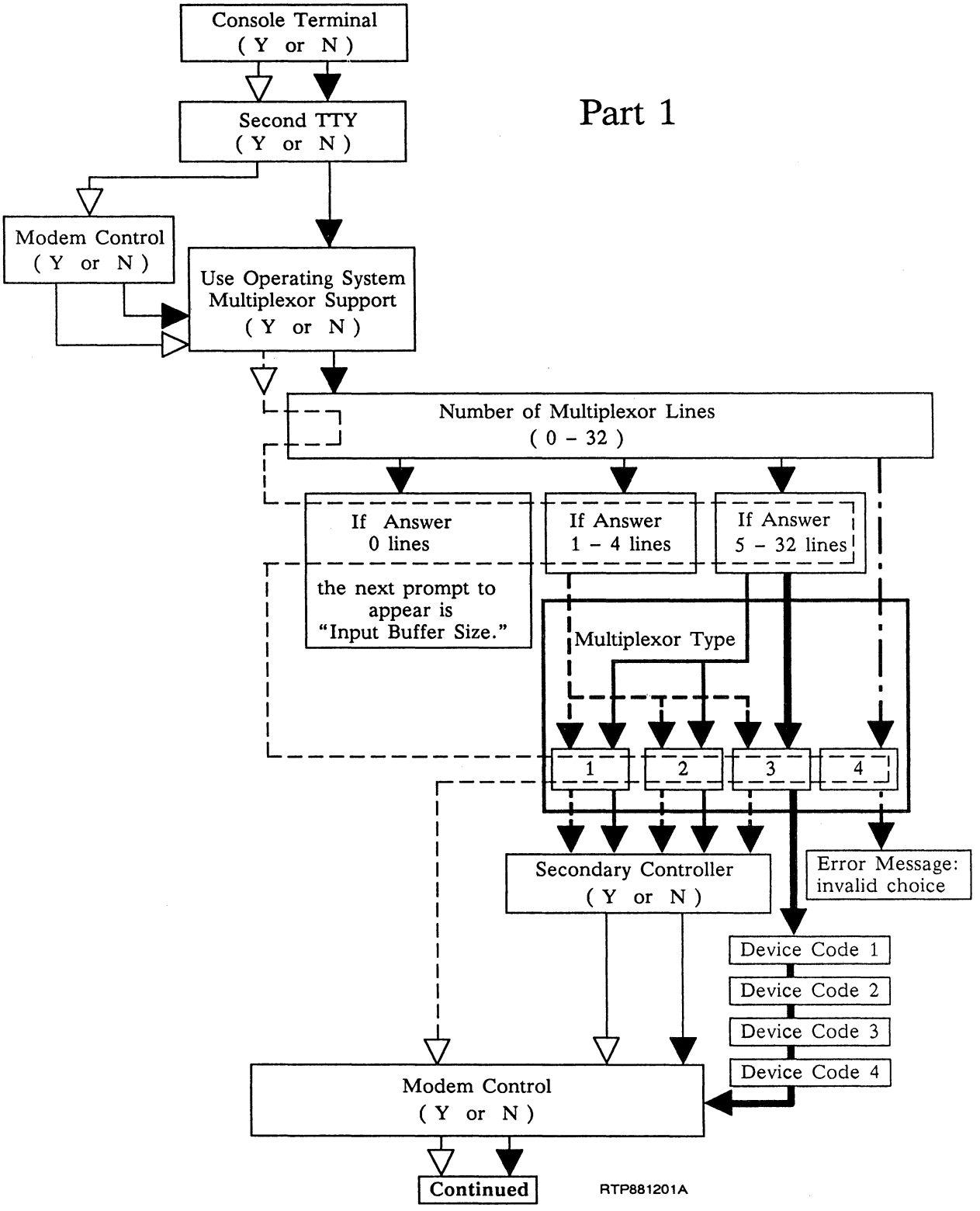


Figure 3-1 Business Basic System Generation (BSG) Dialog

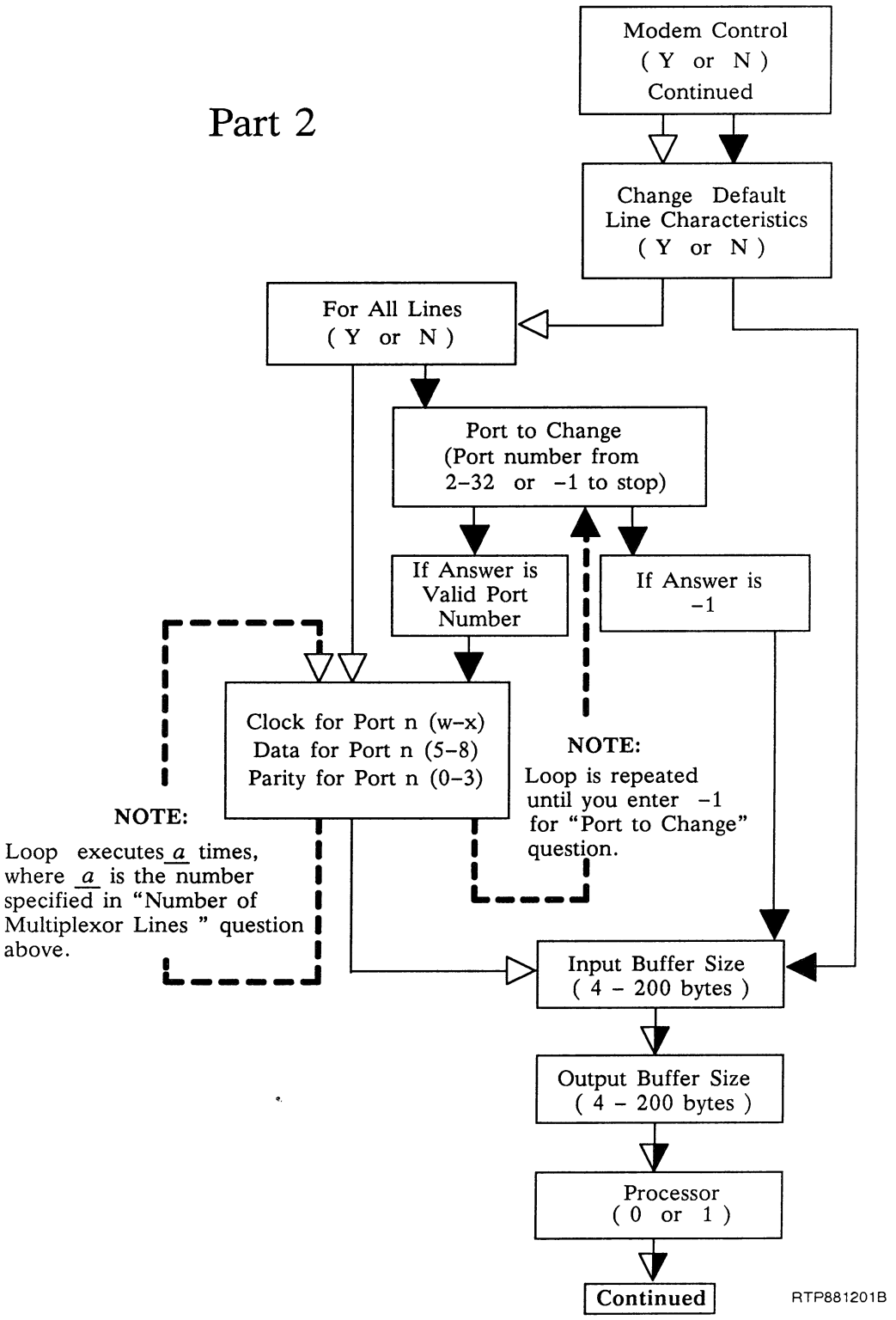


Figure 3-1 Business Basic System Generation (BSG) Dialog *(continued)*

Part 3

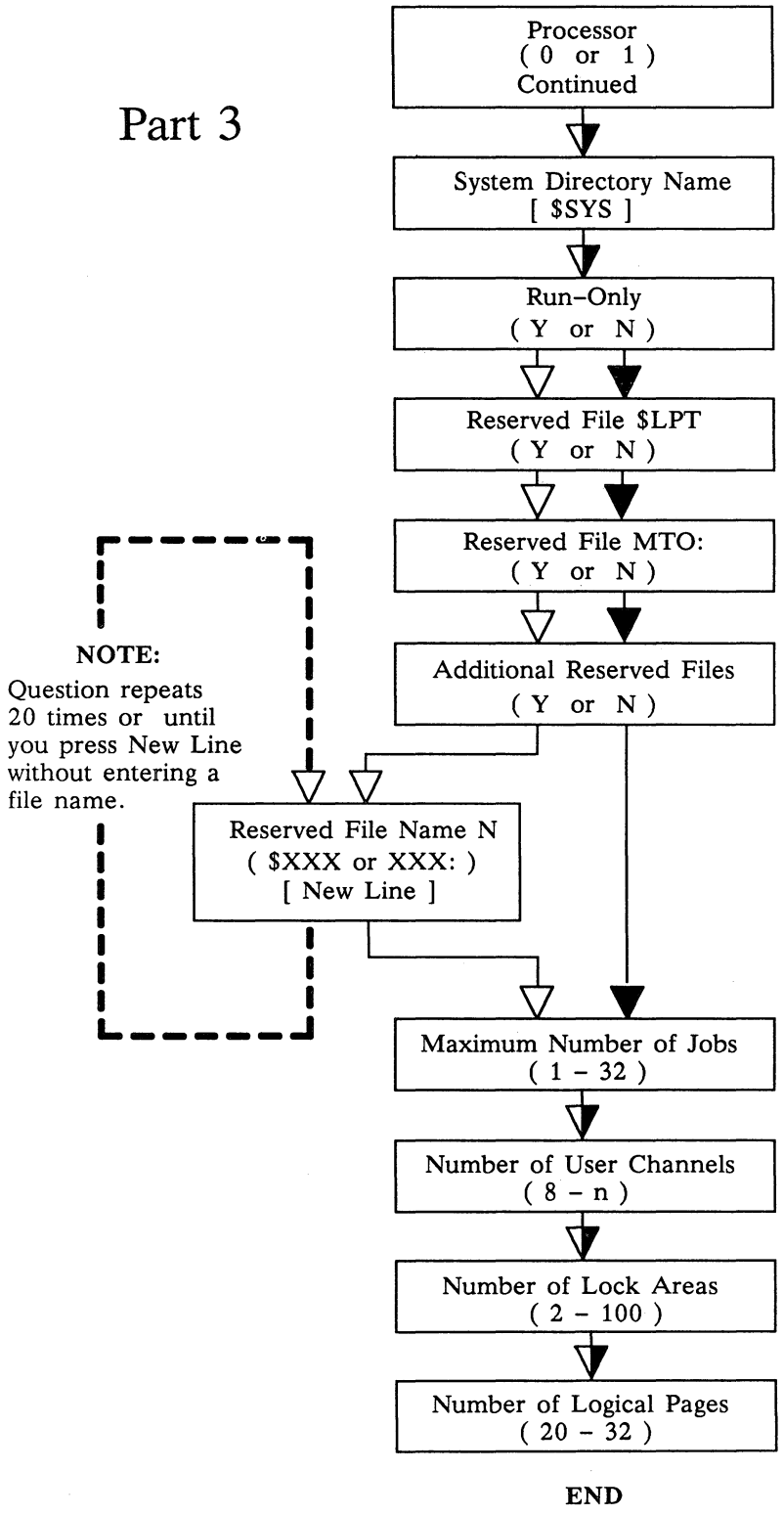


Figure 3-1 Business Basic System Generation (BSG) Dialog *(concluded)*

```
Number of multiplexor lines (0 - 32) [0]?
```

Respond with the number of lines on the multiplexor(s).

Business BASIC multiplexors are not generated or configured into RDOS and are not used by RDOS.

For multiple USAM or ASLM boards, Business BASIC assumes all boards have four lines (even USAM-1 boards). For these boards, you may need to specify a number larger than the actual number of lines. For example, for one USAM-4 and two USAM-1s, request nine lines (that is four lines for the USAM-4, four lines for the first USAM-1, and one more for the second USAM-1).

Line n of a multiplexor maps to Business BASIC port n+2, where multiplexor lines are numbered beginning with 0 (that is, multiplexor line 3 is Business BASIC port 5). If operating system multiplexor support is chosen, enter (n+1), where n is the number of the highest line you want to use. For example, if you want to log on to lines 0, 1, 2, and 3, you must specify 4 here. If you want to log on to lines 1 and 3 only, you must still specify 4 here. If you want to be able to log on to Business BASIC using any line specified in RDOS, respond with the total number of lines specified in RDOS.

If you choose operating system multiplexor support, the number of lines specified in Business BASIC should not exceed the number of lines specified in RDOS.

```
Multiplexor type (1=ALM, 2=ULM, 3=ASLM/USAM, 4=OTHER) (1 - 4) [1]?
```

Respond with the appropriate multiplexor type.

An ALM multiplexor has four clocks with jumpers to set the baud rates. Model numbers for ALMs include 4255, 4256, 4258, 4260, 4340, and 4342. A ULM has 16 predefined clocks. Its model numbers include 4241 and 4225/4227. USAM and ASLM multiplexors have 16 clocks like ULM multiplexors but use a different I/O command set. Their model numbers include 4336 and 4463. Any of these three multiplexor types can be used with Business BASIC support or operating system support.

"OTHER" refers to any other multiplexor type. Included in this category are LACs, DUARTs, and other multiplexors supported by DG/RDOS. These other multiplexor types can be used only with operating system support.

If you answered Y to "Use operating system multiplexor support," the question "Modem control" is now displayed. If you answered N to select Business BASIC multiplexor support, you are now asked:

```
Secondary controller (Y or N) [N]?
```

This question is asked when the multiplexor type is 1 or 2 and when four or fewer

lines are requested for multiplexor type 3. Respond Y if the multiplexor is on the secondary device code. The secondary device code for ALMs and ULMs is 44 and for ASLMs and USAMS is 74. The primary device code is 34 (octal) for all multiplexor types.

If you answer Y here, you cannot generate or configure the multiplexors in RDOS.

The following questions are asked when multiple USAM support is requested by asking for Business BASIC multiplexor support, more than four lines, and multiplexor type 3.

```
Device code 1 (34, 74, 55, 56, -1 [octal]) (-1 - 77) [34]?

Device code 2 (34, 74, 55, 56, -1 [octal]) (-1 - 77) [74]?

Device code 3 (34, 74, 55, 56, -1 [octal]) (-1 - 77) [55]?

Device code 4 (34, 74, 55, 56, -1 [octal]) (-1 - 77) [56]?
```

The first four lines are on device code 1, the next four on device code 2, etc., up to the total number of lines specified previously. Select device codes appropriately for the number of device codes Business BASIC will use, and respond -1 to additional device codes if fewer than four USAMs are to be used. The listed device codes are standard for USAMs; switches on the boards can be set to select any unused device code. If one or more of the boards is a USAM-1, you will have port numbers that cannot be accessed.

Any device code specified here cannot be used by RDOS.

The error message "Invalid type" is displayed if you specify Business BASIC multiplexor support and multiplexor type 4 (regardless of the number of multiplexor lines you specify). Business BASIC multiplexor support is available only for multiplexor types 1, 2, and 3.

Whether you specify Business BASIC or operating system multiplexor support, the following questions are displayed:

```
Modem control (Y or N) [N]?
```

Respond Y if your multiplexor has a modem control option that is installed and that is to be used. Business BASIC's modem support provides automatic logon and detaches a job if the phone connection is broken.

Under Business BASIC multiplexor support, automatic logon and automatic detachment are done by monitoring and manipulating the signals Ring Indicator (RI), Carrier Detect (CD), Request To Send (RTS), and Data Terminal Ready (DTR). When a line is not in use, Business BASIC raises RTS and lowers DTR. When someone calls in, RI is seen. (If your board has a switch that enables ring interrupts,

set the switch **on** for modem lines and **off** for local lines; USAMs and ASLMs have such switches.) Business BASIC then raises DTR and waits several seconds for CD, at which time it simulates pressing Escape, which is normally required for logon and the HELLO program is run. After a user says BYE and hangs up the phone, or if the phone connection is broken at any time, CD drops and Business BASIC waits several seconds before re-setting RTS and DTR to their initial states. If CD is lost while a user is logged on, the job is detached from the line.

```
Change default line characteristics (Y or N) [N]?
```

By default, all multiplexor lines are set to 9600 baud (which is assumed to be clock 0 on ALM multiplexors) and 7 data bits. ALM and ULM multiplexors use mark parity; USAM and ASLM multiplexors use even parity (there is no mark parity setting on these devices). Answer Y if you would like any of your multiplexor lines to be set differently (in other words, if you used characteristics other than these when you generated your operating system). The first multiplexor line is port 2; ports 0 and 1 are reserved for the master and secondary consoles even if they are not requested during the BSG program.

If you answer N to this question, the "Input buffer size" question is displayed next. If you answer Y to this question, the following prompt appears:

```
For all lines (Y or N) [N]?
```

If you answer Y here, you are prompted for information about all the lines in your system, beginning with port 2 (multiplexor line 0). The first multiplexor line is port 2; ports 0 and 1 are reserved for the master and secondary consoles even if they are not requested during the BSG program. If you answer Y here, the question "Clock for port n" is displayed next. If you answer N, you are asked to specify the lines to be changed:

```
Port to change (-1 to stop):
```

Enter the number of one port for which you want to change the line characteristics. This prompt is repeated until you respond with the value -1, which terminates the loop.

If you answered Y to "For all lines," the following set of questions is repeated for each of the port numbers from 2 through the highest port number (specified in the "Number of multiplexor lines" question). If you answered N to "For all lines," these questions are displayed each time you enter a valid port number under "Port to change." Values for w, x, y, and z depend on the multiplexor type and are explained by a message that is displayed before this series of questions.

```
Clock for port n (w - x) [y]?
```

```
Data bits for port n (5 - 8) [7]?
```

```
Parity for port n (0 - 3) [z]?
```

Clocks and corresponding baud rates for multiplexor type 1 depend on how your board is jumpered. Most often, clock 0 is 9600 baud and clock 1 is 600 baud. Typically, clock 2 may be 1200 or 9600 baud and clock 3 may be 110, 150, 300, or 2400 baud.

Clocks and corresponding baud rates for multiplexor type 2, ULMs, are:

0=1	4=134.5	8=9600	12=2400
1=19200	5=200	9=4800	13=300
2=50	6=600	10=1800	14=150
3=75	7=2400	11=1200	15=110

Clocks and corresponding baud rates for multiplexor type 3, ASLMs and USAMs, are:

0=50	4=150	8=1800	12=4800
1=75	5=300	9=2000	13=7200
2=110	6=600	10=2400	14=9600
3=134.5	7=1200	11=3600	15=19200

Parity choices for multiplexor types 1 and 2, ALM and ULM, are:

0 = none
1 = odd
2 = even
3 = mark

Parity choices for multiplexor type 3, ASLM and USAM, are:

0 = odd/disabled
1 = odd/enabled
2 = even/disabled
3 = even/enabled

The following questions are displayed either when you answer N to "Change default line characteristics" or when you complete the above questions on default line characteristics.

```
Input buffer size (bytes) (4 - 200) [20]?
```

Respond with the size, in bytes, that you desire for this buffer.

Procedure

If your input buffer is not large enough, characters can be lost on input by a fast typist or during type-ahead while a program is running.

```
Output buffer size (bytes) (4 - 200) [20]?
```

Respond with the number of bytes that the system will hold in the buffer during output.

The larger this buffer, the less frequently Business BASIC must flush the output buffer, and the more time Business BASIC has available for other processing. If your output buffer is too small, performance can be affected. For example, if you set this buffer size to 10 bytes, then Business BASIC must suspend its processing of program code each time it has 10 bytes to write to a screen or file. On the other hand, when a printer is attached to a Business BASIC multiplexor line, too large a value can cause overflow of the printer's hardware buffer.

If your output buffer is a different size from your input buffer, characters that are input may not be echoed properly.

```
Processor (0=Single, 1=Secondary) [0]?
```

Respond 1 if the system is to run in the foreground while another system (that typically would have chosen 0 for this question) is running in the background. Each system must have its own system directory to avoid conflicts over BASIC.PS, the push file.

If you will be running two Business BASIC systems on the same machine, note that they cannot share the $SYS (or $SY3) directory. This is true because each system must have its own BASIC.PS file; this file is used to store a program while a user swaps to another program. Thus, for one of the two systems, answer 1 to this question.

```
System directory name [$SYS]?
```

If you answer 1 to the processor question, choose a four-character name that starts with $ for the system directory (for example, $SYF). Before bringing up the system, create a directory with this name. This directory should contain HELLO, BASIC.ER, and a system program library if desired.

```
Run-only (Y or N) [N]?
```

Respond Y for a run-only system. This restricts the loading of a portion of Business BASIC's syntax and code listing sections. Thus, you can use only the RUN command, and the prompt changes to * RUN. In a run-only system the ENTER statement or command does not work even if you include it in programs. Use of

a run-only system can reduce the interpreter's size, providing more space for user programs.

```
Reserved file $LPT (Y or N) [Y]?
```

Answer Y if the system line printer is to be included in the reserved file table. This is required if you wish to use $LPT as a filename in an OPEN FILE statement.

Since devices cannot be opened exclusively in RDOS, reserved files provide a mechanism for Business BASIC users to have exclusive access to devices.

```
Reserved file MT0: (Y or N) [Y]?
```

Answer Y if magnetic tape unit 0 is to be included in the reserved file table.

Tape units are the only filenames containing colons (:) that are allowed in OPEN FILE statements, unless the user status bit 8 of U.S2 is set to allow files with directory specifiers. (See the RDOS section "Directory, File and Link Access" in Chapter 5, "Security.")

```
Additional reserved files (Y or N) [N]?
```

Answer Y if any additional names are to be added to the reserved file table. The following question is asked until you respond with a carriage return:

```
Reserved file name n []?
```

The value of n can range from 1 to 20. Respond with a reserved filename that is exactly four characters long and either begins with a dollar sign ($) or ends with a colon (:). Such filenames are rejected by OPEN FILE unless they are in the reserved file table. If the true name of the reserved file does not follow these conventions, create a link and define the link as the reserved file; for example, for a secondary system printer $LPT1, respond $LPA to this question and link $LPA to $LPT1.

```
Maximum number of jobs (1 - 32) [1]?
```

Respond with the maximum number of jobs that can run at one time. The Business BASIC execution command actually controls the number of jobs running, up to the number you specify here as the maximum. The number of jobs must be able to simultaneously accommodate user terminal jobs and detached jobs, including spooler jobs.

The number chosen here sets an upper limit on what you can specify with the local /J switch when you execute Business BASIC. The maximum number of concurrent jobs depends upon available memory and the size of the operating system, the Business BASIC interpreter, and the user program and data area.

Being able to fit a certain number of users on a system does not guarantee good performance. Program design is an important consideration. The maximum available user program and data space decreases as the number of jobs (at both BSG and execution time) increases.

```
Number of user channels (8 - n) [8 * number of jobs]?
```

Respond with the total number of channels that can be opened by all Business BASIC programs. Business BASIC will not run if more channels are generated into it than are generated into RDOS.

The maximum number of channels a user can have open at one time via OPEN FILE is 16. However, it is possible at runtime to require more than 16 channels per user through the use of user program libraries and files opened via STMCs on "negative" channels. For information on negative channels, see the "STMCs in RDOS Business BASIC" section of Chapter 7.

```
Number of lock areas (2 - 100) [2 * number of jobs]?
```

Respond with the total number of LOCKs that can be active simultaneously. Having too few LOCK areas results in LOCK requests waiting for an available area; having too many can affect performance because each area must be checked for conflicts on every LOCK attempt.

```
Logical pages (20 - 32) [32]?
```

The default answer causes Business BASIC to allocate a full 32 pages (64 Kbytes) of logical memory upon execution, maximizing the available user program and data space. Any remaining memory (that is not used by the operating system or the Business BASIC interpreter and overlays) is available for user program and data areas in extended memory. The maximum user program and data space (i.e., the maximum local /M value at execution time) depends on several BSG options; if your programs are smaller than your maximum /M value, you can reduce the answer to this question and thus free some memory. This memory could be used to increase the available extended memory (making it possible to run additional jobs), or it could be given to the other ground via the operating system's CLI command SMEM.

Total memory required by a Business BASIC system (not counting memory for the operating system or for another ground) can be determined by the following formula:

total KB = [logical pages * 2] + 24 + [/J value - 1] * [/M value + 2]

After you respond to the last question of the BSG dialog (assuming you did not use the /N global switch), BSG terminates Business BASIC and returns you to RDOS, and RDOS executes the commands in the CLI.CM file created by BSG. If no MAC.PS exists in the BASICGEN (or BASICGEN3) directory, the commands in the CLI.CM file execute the BUILDBBPS macro, which creates several links to the master directory and constructs a MAC.PS. by executing the BUILDBBPS macro. In

any case, the commands in systemname.CM are then executed; systemname.SR and SY.SR are assembled to produce systemname.RB and systemname.LS using MAC; and systemname.SV, systemname.OL, and systemname.LM are created using RLDR. If you want to execute the system from a directory other than BASICGEN (or BASICGEN3), you can move or link to the .SV and .OL files. You can use the MAC.PS created by BUILDBBPS to assemble user subroutines or modified CRT modules, and you can modify BUILDBBPS.MC to include other parameter files your modules use.

Including Assembly Language Subroutines in RDOS Business BASIC

You can increase Business BASIC functions with assembly language subroutines. To use them, you generate a version of Business BASIC that includes an interface for the subroutines. Users then access these subroutines with the UCALL statement or command.

The steps for implementing assembly language subroutines are:

1) Put assembly subroutines in USERSUBS.SR.

2) Use MAC to produce USERSUBS.RB.

3) Use BSG to produce a new system.

The module USERSUBS is automatically loaded into any Business BASIC interpreter. The default USERSUBS module supplied on the release media defines no routines. To implement a UCALL, you should edit and assemble the file USERSUBS.SR in the BASICGEN (or BASICGEN3) directory and then generate a system. Whenever a system is built, the USERSUBS.RB file in the BASICGEN (or BASICGEN3) directory is included.

The USERSUBS.SR file on the release media contains a commented-out sample user subroutine employing several macros that simplify the writing of user subroutines and are defined in parameter files supplied with the release media. Use a permanent symbol file that includes these parameter files when the USERSUBS module is assembled. The BUILDBBPS macro builds the proper MAC.PS file. BUILDBBPS is executed the first time a complete BSG is performed, so a MAC.PS file usually is present in the BASICGEN (or BASICGEN3) directory. The other parameter files and macros are in the BASICGEN and BASICGEN3 directories; the macros also use operating system parameter files that must be in your master directory.

Numeric and string variables and expressions can be the arguments for a UCALL; however, only single elements of numeric arrays may be used. All variables must be dimensioned or initialized before the UCALL.

A UCALL has up to eight arguments. The start of a UCALL and its arguments are defined with the ASMDEF macro. The format is:

ASMDEF number,argument-1,...,argument-n

where:

number — Is the subroutine number -- for example, 10. is the subroutine number for UCALL 10. The default radix for MAC is octal; thus the decimal point indicates a decimal number.

argument-n — Is the type of the nth argument. Possible values are:

SEX	String expression
NEX	Numeric expression
SV	String variable
NV	Numeric variable

The location immediately after the ASMDEF statement should be the start of the subroutine.

After all UCALLs are defined with ASMDEF, use the DEFDMP macro to create the subroutine table that the interpreter refers to during the evaluation of arguments in preparation for the call. This table must be at location SBRTB. Thus, you should end your USERSUBS module with:

```
SBRTB: DEFDMP
```

Business BASIC passes two pointers to the subroutine. The first is a word pointer to an argument list; this pointer is in accumulator 2 and also in the ZREL location TR1. Note that the pseudo-op .EXTD must be coded if you refer to TR1 or any other special ZREL locations. The second pointer is the return address, which is used at the end of your UCALL. To return, either JMP @TR15 (and code .EXTD TR15), or JMP to the word immediately following the argument list.

The argument list contains a one-word entry for each argument. For numeric arguments, the argument list entry is a word pointer to the high-order 16 bits of the value of the numeric variable or expression. Numeric values are stored internally in the following manner:

Double Precision	**Triple Precision**
High order	High order
Low order	Middle order
	Low order

For string expressions, the argument list entry is a word pointer to a two-word descriptor string that contains the following:

- Byte address of the first byte of the string
- Byte length of the string

For string variables, the argument list entry is a word pointer to a four-word descriptor string that contains the following:

- Byte address of the first byte of the string or substring
- Byte length of the string or substring
- Maximum byte length of the string or substring
- Word address of the current length of the string

For example, if A$ is dimensioned to 10 bytes and is completely full, and the argument to a UCALL is A$[6,10], the second and third words of the descriptor string both contain the value 5. The fourth word is a pointer to a location that contains the value 10.

Business BASIC statements like PRINT and WRITE cannot use information beyond the current length of a string. If a result is to be returned in a string variable supplied in the UCALL, then the string variable should be null-filled before the UCALL is issued, so that the string variable is allocated. The UCALL should update the current length of the string after the information has been stored in the string.

For UCALL temporary storage areas, use:

- ZREL locations TR0 through TR15. They must be declared with .EXTD. Remember that TR1 and TR15 contain special pointers.
- ECLIPSE stack instructions. If you make changes to the stack, you must restore the original order of items in the stack. Use an EJSR RESBB instruction to reset the stack to its initial state.
- Local storage locations (if the code need not be re-entrant).

To load constants into accumulators, use:

- The ELEF instruction
- The literal facility of the assembler
- Locally hard-coded constants

Set the NSWP bit in the User Status Table before using system calls that use buffers in the program and data area or other special areas (like the TR registers) that cannot be mapped. (Refer to PARBU.SR in the BASICGEN or BASICGEN3 directory.)

You can use USERSUBS as resident code, or you can load USERSUBS in an overlay by editing the RLDR line. When you use overlays, the overlay must not cross a 4-Kbyte boundary.

Test your UCALLs before they are linked into a production system. Because of the level on which a UCALL works, a UCALL with errors can cause system crashes and other serious faults. UCALLs aren't always transportable from revision to revision or from operating system to operating system. Also, enhancements and changes to scheduling and memory management algorithms can force the revision of UCALLs.

Sample System Generation Dialogs

Below is a sample system generation appropriate for a DESKTOP GENERATION® system with a USAM multiplexor.

```
Console terminal (Y or N) [Y]? Y
Second TTY (Y or N) [N]? N
Use operating system multiplexor support (Y or N) [N]? Y
Number of multiplexor lines (0 - 32) [0]? 4
   Multiplexor type (1=ALM,2=ULM,3=ASLM/USAM,4=OTHER)(1-4) [1]? 3
   Secondary controller (Y or N) [N]? N
   Modem control (Y or N) [N]? N
   Change default line characteristics (Y or N) [N]? N
Input buffer size (bytes) (4 - 200) [20]? 20
Output buffer size (bytes) (4 - 200) [20]? 20
Processor (0=Single, 1=Secondary) (0 - 1) [0]? 0
Run-only (Y or N) [N]? N
Reserved file $LPT (Y or N) [Y]? Y
Reserved file MT0: (Y or N) [Y]? N
Additional reserved files (Y or N) [N]? N
Maximum number of jobs (1 - 32) [1]? 10
Number of user channels (8 - 250) [80]? 80
Number of lock areas (2 - 100) [20]? 20
Logical pages (20 - 32) [32]? 32
```

Below is a sample system generation appropriate for a system with an ALM multiplexor. In this example, several of the lines are changed to use clock 2 instead of the default clock 0.

```
Console terminal (Y or N) [Y]? Y
Second TTY (Y or N) [N]? N
Use operating system multiplexor support (Y or N) [N]? N
Number of multiplexor lines (0 - 32) [0]? 8
  Multiplexor type (1=ALM,2=ULM,3=ASLM/USAM,4=OTHER)(1-4) [1]? 1
  Secondary controller (Y or N) [N]? N
  Modem control (Y or N) [N]? N
  Change default line characteristics (Y or N) [N]? Y
    Clock for port 2 (0 - 3) [0]? 0
    Data bits for port 2 (5 - 8) [7]? 7
    Parity for port 2 (0 - 3) [3]? 3
    Clock for port 3 (0 - 3) [0]? 0
    Data bits for port 3 (5 - 8) [7]? 7
    Parity for port 3 (0 - 3) [3]? 3
    Clock for port 4 (0 - 3) [0]? 2
    Data bits for port 4 (5 - 8) [7]? 7
    Parity for port 4 (0 - 3) [3]? 3
    Clock for port 5 (0 - 3) [0]? 2
    Data bits for port 5 (5 - 8) [7]? 7
    Parity for port 5 (0 - 3) [3]? 3
    Clock for port 6 (0 - 3) [0]? 2
    Data bits for port 6 (5 - 8) [7]? 7
    Parity for port 6 (0 - 3) [3]? 3
    Clock for port 7 (0 - 3) [0]? 2
    Data bits for port 7 (5 - 8) [7]? 7
    Parity for port 7 (0 - 3) [3]? 3
    Clock for port 8 (0 - 3) [0]? 0
    Data bits for port 8 (5 - 8) [7]? 7
    Parity for port 8 (0 - 3) [3]? 3
    Clock for port 9 (0 - 3) [0]? 0
    Data bits for port 9 (5 - 8) [7]? 7
    Parity for port 9 (0 - 3) [3]? 3
Input buffer size (bytes) (4 - 200) [20]? 20
Output buffer size (bytes) (4 - 200) [20]? 20
Processor (0=Single, 1=Secondary) (0 - 1) [0]? 0
Run-only (Y or N) [N]? N
Reserved file $LPT (Y or N) [Y]? Y
Reserved file MT0: (Y or N) [Y]? N
Additional reserved files (Y or N) [N]? N
Maximum number of jobs (1 - 32) [1]? 20
Number of user channels (8 - 251) [160]? 160
Number of lock areas (2 - 100) [40]? 40
Logical pages (20 - 32) [32]? 32
```

Below is a sample system generation appropriate for a system with a ULM multiplexor. In this example, an additional reserved file named $LPA is generated.

```
Console terminal (Y or N) [Y]? Y
Second TTY (Y or N) [N]? N
Use operating system multiplexor support (Y or N) [N]? N
Number of multiplexor lines (0 - 32) [0]? 8
   Multiplexor type (1=ALM,2=ULM,3=ASLM/USAM,4=OTHER)(1-4) [1]? 2
   Secondary controller (Y or N) [N]? N
   Modem control (Y or N) [N]? N
   Change default line characteristics (Y or N) [N]? N
Input buffer size (bytes) (4 - 200) [20]? 50
Output buffer size (bytes) (4 - 200) [20]? 50
Processor (0=Single, 1=Secondary) (0 - 1) [0]? 0
Run-only (Y or N) [N]? N
Reserved file $LPT (Y or N) [Y]? Y
Reserved file MT0: (Y or N) [Y]? N
Additional reserved files (Y or N) [N]? Y
   Reserved file name 1 []? $LPA
   Reserved file name 2 []?
Maximum number of jobs (1 - 32) [1]? 12
Number of user channels (8 - 250) [96]? 96
Number of lock areas (2 - 100) [24]? 24
Logical pages (20 - 32) [32]? 32
```

The following generation example is for a system that has operating system multiplexor support, runs on a DESKTOP GENERATION computer with a 4-line USAM, has a printer but no tape drive, and supports up to 4 jobs:

```
Console terminal (Y or N) [Y]? Y
Second TTY (Y or N) [N]? N
Use operating system multiplexor support (Y or N) [N]? Y
Number of multiplexor lines (0 - 32) [0]? 4
   Multiplexor type (1=ALM,2=ULM,3=ASLM/USAM,4=OTHER)(1-4) [1]? 3
   Modem control (Y or N) [N]? N
   Change default line characteristics (Y or N) [N]? N
Input buffer size (bytes) (4 - 200) [20]? 20
Output buffer size (bytes) (4 - 200) [20]? 20
Processor (0=Single, 1=Secondary) (0 - 1) [0]? 0
Run-only (Y or N) [N]? N
Reserved file $LPT (Y or N) [Y]? Y
Reserved file MT0: (Y or N) [Y]? N
Additional reserved files (Y or N) [N]? N
Maximum number of jobs (1 - 32) [1]? 4
Number of user channels (8 - 251) [96]? 37
Number of lock areas (2 - 100) [24]? 8
Logical pages (20 - 32) [32]? 32
```

The sample system generation shown below is appropriate for a DG/500 computer with 16 lines. It uses operating system multiplexor support.

```
Console terminal (Y or N) [Y]? Y
Second TTY (Y or N) [N]? N
Use operating system multiplexor support (Y or N) [N]? Y
Number of multiplexor lines (0 - 32) [0]? 16
   Multiplexor type (1=ALM,2=ULM,3=ASLM/USAM,4=OTHER)(1-4) [1]? 4
   Modem control (Y or N) [N]? N
   Change default line characteristics (Y or N) [N]? N
Input buffer size (bytes) (4 - 200) [20]? 20
Output buffer size (bytes) (4 - 200) [20]? 20
Processor (0=Single, 1=Secondary) (0 - 1) [0]? 0
Run-only (Y or N) [N]? N
Reserved file $LPT (Y or N) [Y]? Y
Reserved file MT0: (Y or N) [Y]? Y
Additional reserved files (Y or N) [N]? N
Maximum number of jobs (1 - 32) [1]? 5
Number of user channels (8 - 251) [96]? 46
Number of lock areas (2 - 100) [24]? 10
Logical pages (20 - 32) [32]? 32
```

End of Chapter

Chapter 4
Executing Business BASIC

This chapter describes how to execute the Business BASIC you created at system generation. At execution you can use switches and arguments to further modify your system. Before executing your system, study carefully the options described below.

Executing AOS Business BASIC

To execute AOS Business BASIC, put $SYSLIB or $SYSLIB3 and the parent directory (:UTIL:BBASIC in our examples) in the user's search list and enter the execution command line.

```
) SEARCHLIST :UTIL:BBASIC :UTIL:BBASIC:$SYSLIB [!SEA] :UTIL :

) EXECUTE systemname[/C] [/D] [/E] [/N] [/P=blocks]
[/Q=default queue] [/S=program name] [/T] [/U] [/W] [/X]
```

A few of the Business BASIC CLI commands (AOS, STAT, and VFU) invoke AOS programs (CLI.PR, PED.PR, and FCU.PR, respectively). To execute these commands, add the directories containing the .PR files to your user's search list. For you to use these commands, your AOS profile must allow you to create son processes.

Selecting Options at Execution

You can attach several global switches to BBASIC (or to systemname) to modify your Business BASIC system. The switches allowed are:

Switch	Description
/C	Allows you to use the Business BASIC CLI.
/D	Enables a Ctrl-C Ctrl-B sequence to abort a process or a Ctrl-C Ctrl-E sequence to abort a process and create a break file. This interrupt can alter your terminal's characteristics. To save these characteristics, do a PUSH prior to executing Business BASIC. Issue a POP command after you leave Business BASIC to restore your terminal's previous characteristics.
/E	Enables terminal types 8 and 9. For this switch to have the desired effect, you must have included terminal types 8 and 9 when you generated Business BASIC. Otherwise, your screen will not be cleared when Business BASIC is executed, and you will not have the capabilities of terminal types 8 and 9.
/N	Limits communication to the record lock server (RLS2) to IPCs rather than allowing local locks of the shared database. This locking method consumes more CPU cycles. However, in the 16-bit version, this switch frees 2 Kbytes of program space.
/P=blocks	Defines the user program size as the number of 512-byte blocks. Set blocks to a multiple of four. The minimum size is 16 blocks. The default size is the maximum available blocks.
/Q=default-queue	Sets the default output queue. If not set, the default queue is @LPT.
/S=program-name	Starts the Business BASIC program named as soon as the user logs on. Program-name must be a program file that was saved or replaced.
/T	Forces Business BASIC to log the user off after 10 minutes of inactivity.
/U	Disables the default 10-minute bell and the automatic flushing of the input buffer. It also disables the /T switch (see description above).
/W	Passes the user through HELLO to a Business BASIC program without allowing the interactive use of Business BASIC. When the program terminates, a BYE is forced and the user is logged off Business BASIC. This switch is used with the /S=program-name switch.

Switch	Description
/X	Stops allocation of a shared page for use by the ISAM routines when Business BASIC is dealing with 512- or 2048-byte shared page indexes. `Error 89 - Illegal file type` error is returned when /X is selected and an ISAM statement on a file opened in shared mode (4 or 5) is executed. In the 16-bit version, if no shared mode ISAM statements are used (exclusive use of INFOS® II interface, for example), then an additional 2 Kbytes of space is provided for your program and data.

Methods of Execution

The default system name is BBASIC, the name supplied with the .PR and .OL (16-bit only) files in the Business BASIC package. If you generated a system with another name, substitute your system name in the following examples.

1) Log on to AOS and enter:

```
) SEA :UTIL:BBASIC :UTIL:BBASIC:$SYSLIB [!SEA] :UTIL :
) XEQ BBASIC/C
```

This puts you in Business BASIC and permits access to the Business BASIC CLI.

2)
```
) SEA :UTIL:BBASIC :UTIL:BBASIC:$SYSLIB [!SEA] :UTIL :
) CHAIN BBASIC/U
```

This replaces the AOS CLI program with the Business BASIC program and allows unlimited time for input. When you exit Business BASIC, you will be logged off Business BASIC and AOS.

3) Set up a macro, BBASIC.CLI, as follows:

```
PUSH
SEARCHLIST :UTIL:BBASIC :UTIL:BBASIC:$SYSLIB [!SEA] :UTIL :
XEQ BBASIC/C
POP
```

Execute Business BASIC by entering the command: `BBASIC`.

4) You can log on to Business BASIC directly if you use the same macro as in the previous example, but define BBASIC.CLI as the initial IPC file in your AOS profile. Use the operating system PREDITOR program to enter the name of the macro created in the previous example as the initial IPC file.

5) Use the AOS CLI command process to create a new process running Business BASIC.

```
) SEA :UTIL:BBASIC :UTIL:BBASIC:$SYSLIB [!SEA] :UTIL :
) PROCESS/BLOCK/DEFAULT/IOC BBASIC/C/D
```

When you leave Business BASIC, you are returned to the AOS CLI.

6) You can vary the procedure above to give a batch facility.

```
) SEA :UTIL:BBASIC :UTIL:BBASIC:$SYSLIB [!SEA] :UTIL :
)PROC/BLO/DEF/IOC/INPUT=filename1/OUTPUT=filename2
BBASIC/C/D
```

The input file, filename1, is a text file containing Business BASIC commands, and the output file, filename2, receives the output that normally is displayed on the screen. Filename2 must exist before the PROC command is issued.

This method puts screen output from jobs that have run in a file that can be examined for errors later.

You can also use this with the CLI command QBATCH. Create a text file containing the PROC command and enter: QBATCH text-file-name.

7) You can also use the following example to create a macro that can be executed with the CLI's QBATCH command.

The example file SAVEFILES contains the following:

```
SEA :UTIL:BBASIC :UTIL:BBASIC:$SYSLIB [!SEA] :UTIL :
XEQ/M BBASIC
ENTER "FILE1.LS"
SAVE "FILE1"
NEW
ENTER "FILE2.LS"
SAVE "FILE2"
NEW
ENTER "FILE3.LS"
SAVE "FILE3"
BYE
)
```

At the CLI prompt, you enter:

```
)QBATCH/NOTIFY SAVEFILES
```

This causes each line of SAVEFILES to be passed to the AOS CLI for execution. The line `XEQ/M BBASIC` causes Business BASIC to be executed; the /M switch enters all the information before the right parenthesis as input

to Business BASIC. Business BASIC enters and saves three files and is terminated by the BYE statement. The /NOTIFY sends a message when the job is complete.

Logging On and Off Business BASIC

When you execute Business BASIC, you are automatically logged on without any additional account name or password prompts.

Enter BYE to log off.

If you logged on with a BBASIC.CLI macro as created above, the Business BASIC statement or command BYE pops you up one AOS level. If you logged on with XEQ BBASIC from the CLI, a BYE puts you back in the CLI and terminates the son process that was executing BBASIC. If you logged on with CHAIN BBASIC from the CLI, a BYE logs you off the AOS system.

The Business BASIC CLI command `!AOS` creates another son process that runs the AOS CLI. You can then use EXECUTE or CHAIN BBASIC to create another son process running Business BASIC. In this second-level Business BASIC, BYE puts you back one level to the son process running the AOS CLI.

If you have disabled IKEYs (STMA 6,5) and you are running a program that never stops, shut down Business BASIC with a Ctrl-C Ctrl-B key sequence. (You must have used the /D switch at execution to use a Ctrl-C Ctrl-B.) This interrupt can alter your terminal's characteristics. To save these characteristics, do a PUSH prior to executing Business BASIC. A POP then restores your terminal's previous characteristics. Refer to the AOS CLI documentation for more information.

Push Space

When the SWAP statement or command is executed, the program in working storage and information about its current state are written to a file named BASIC.PS (the push file). Business BASIC creates BASIC.PS in the directory where you execute Business BASIC. BASIC.PS is sometimes called the swap file. Business BASIC sets up the AOS push file automatically, but push file users must have WRITE access to the directory where Business BASIC has been executed.

When Business BASIC is executed, it creates and opens the push file, using an algorithm involving the PID number to generate a unique filename. After the file is opened, it is immediately deleted. The actual physical deletion is not performed until the file is closed, but the file is not visible to the FILESTATUS command during Business BASIC execution.

The push file depth is limited by the maximum sizes of any control point directories above the directory containing the push file and the available blocks on the disk.

Push Space

System and User Program Libraries

When Business BASIC is executed, it opens the system program library (BASIC.PL), if you have built one. For more information, see Program Library Builder in the AOS section of the "Utilities" chapter.

Executing RDOS Business BASIC

Execute RDOS Business BASIC by entering the name of the system you created with BSG along with any desired switches and arguments. You must execute RDOS Business BASIC in a directory where systemname.OL and systemname.SV reside or in a directory where links to these files exist. If $SYS and $LIB are not in your current partition when you execute Business BASIC, you must initialize $SYS and $LIB ($SY3 and $LIB3 for triple precision). If you created a system directory for the foreground, you must initialize it in the manner just described for $SYS.

The format for executing Business BASIC is:

systemname[/B][/D][/K][/S][/U][/Z] [jobs/J] [line/B]... [program-size/M] [startfile/S[/n]]...

Selecting Options at Execution

Use these switches and their arguments to select RDOS Business BASIC features.

Global Switches

Switch	Description
/B	Displays a banner for each job specified at execution. This switch is applicable only when operating system multiplexor support has been selected. Under Business BASIC multiplexor support, a banner is automatically displayed for each job specified at execution.
/D	Enables RDOS keyboard interrupts. This switch is recommended only for single-user systems or systems where the master console is in a secure environment. If the RDOS keyboard interrupts are enabled, pressing the interrupt keys on the operating system console brings down the Business BASIC system. Since you can always use KILL, QUICKILL, or STMB 17 to bring down the Business BASIC system, do not use the /D global switch except in unique situations. In RDOS, Ctrl-C Ctrl-A causes an interrupt and Ctrl-C Ctrl-B causes a break file.

Switch	Description
/K	Brings up Business BASIC with the kill flag set. Thus, HELLO does not allow nonprivileged users to sign on. This allows any critical start of day processing done by the STARTUP (/S) facility to run without competition or concern for shared data. However, the kill flag prevents the Business BASIC Spooler from running. Therefore, the STARTUP stream must clear this flag before the Spooler is started and normal operation can begin.
/S	Automatically execute the system utility program STARTUP. For more details on STARTUP, refer to the "Automatic Job Execution" section in this chapter.
/U	Disables the default 10-minute bell and buffer flush on input. It also disables the option that forces the system to log you off after 10 minutes of inactivity. If you specified the forced logoff option with the timeout field of the ACCOUNTING file, it is overridden by the /U switch.
/Z	Allows all users to use privileged commands like STMB, STMC, STMD, and DIR and to list programs that have had STMB 16 set.

Local Switches

Switch	Description
jobs/J	Defines the number of jobs that can run simultaneously in your Business BASIC system. The fewer jobs allowed, the more memory exists as program space. You need one page of memory (2048 bytes) for table and buffer areas for approximately every four users. The default number of jobs is one.
line/B	Defines a line to which the banner is sent when Business BASIC is executed. This switch only applies when operating system multiplexor support is selected. When Business BASIC multiplexor support is selected, the Business BASIC default is to send banners to all lines. To send a banner to several lines, use multiple line/B arguments.

Switch	Description
program-size/M	Defines the amount of user program and data space in kilobytes. The default is 24 Kbytes of program space; if this is impossible, or if a specified /M value is too large, an initialization error occurs. If you choose an odd number, the size is rounded up to the next even number.
startfile/S[/n]	Startfile is the name of a file containing Business BASIC CLI commands that will be started by STARTUP. The optional /n switch, where n is a number, is used to indicate the number of jobs of this name that are to be executed. The default n value is 1. For this switch to execute properly, you must also specify the global /S switch.

Methods of Execution

The following are Business BASIC execution examples. The systemname is BBASIC.

1) `BBASIC/D 32/M`

   This command executes the file BBASIC.SV to bring up the Business BASIC system and allows Business BASIC to execute only one job. In this case, keyboard interrupts can abruptly bring down the Business BASIC system, and 32 Kbytes of program space is requested.

2) `BBASIC 5/J`

   This command brings up Business BASIC and allows five jobs to run simultaneously on the system.

3) `BBASIC/S 10/J SPOOLER/S/2`

   When you bring up Business BASIC, the command line can have the /S switch to cause STARTUP to automatically execute the commands in SPOOLER.JB. Thus, the command brings up Business BASIC with ten jobs; two will be running spoolers.

4) Using an editor or the VACUUM utility described in the RDOS section of the "Utilities" chapter in this manual, set up a macro, START.MC, which contains the following:

```
DIR DZ0:$LIB;GDIR;CLEAR/A/V/D;RELEASE $LIB
DIR DZ0:$SYS;GDIR;CLEAR/A/V/D;RELEASE $SYS
```

```
DIR DZ0:$SPL;GDIR;CLEAR/A/V/D;RELEASE $SPL
DIR DZ0;GDIR;CLEAR/A/V/D
BBASIC/S 5/J SPOOLER/S/2
```

This macro is for a system that was loaded in the master partition, DZ0. The files BBASIC.SV and BBASIC.OL were moved to DZ0 after they were generated. To execute this macro, enter START at the RDOS CLI prompt. The START macro locates and clears the devices and the use counts of files that were open at the time of a system crash. User directories can be added to the macro, and it can be used on a regular basis even if the system does not crash.

When you execute Business BASIC, the message

```
DGC BBASIC x.xx
```

appears on the screen.

5) `BBASIC/B 16/J`

This command brings up Business BASIC with 16 jobs available to run simultaneously. A banner is sent to every terminal line generated into the system.

6) `BBASIC/U 3/B 4/B 7/B 16/J`

This command brings up Business BASIC with 16 jobs available to run simultaneously. A banner is sent only to lines 3, 4, and 7. The 10-minute time-out bell and the buffer flush are disabled.

Logging On and Off Business BASIC

For information about logging on to Business BASIC, see Chapter 5, "Security."

To log off Business BASIC, use the BYE command.

To terminate Business BASIC, use KILL or QUICKILL. They are explained in the RDOS section of the "Utilities" chapter in this manual.

Push Space

When the SWAP statement or command is executed, the program in working storage and information about its current state are written to a file named BASIC.PS (the push file). You must have sufficient space in $SYS:BASIC.PS for the SWAP statement to work. When Business BASIC is executed, it tries to create the push file as a random file; however, if a BASIC.PS exists, it is used. If BASIC.PS is a contiguous file, the SWAP statement performs faster. You can use the RDOS CLI command CCONT to create a contiguous BASIC.PS file with a size specified by this formula:

size = [program-size + 1] * maximum-swaps * maximum-jobs

where:

program-size = maximum program size in 512-byte blocks. (This value is double the program size value defined with the /M switch at execution.)

maximum-swaps = maximum number of nested SWAP statements. (You should use maximum-swaps = 5 if you use CLI commands, because CLI commands often result in several levels of SWAP statements.)

maximum-jobs = maximum number of jobs.

Automatic Job Execution

When you execute Business BASIC from the RDOS CLI, you can use the /S global switch to automatically execute STARTUP programs in the $SYS ($SY3) directory. You can modify STARTUP to do any type of system housekeeping.

STARTUP looks for filenames with the /S local switch in the RDOS command line you issued to execute Business BASIC. Put each of the named files in $SYS or $SY3 with a .JB extension. Do not specify the .JB extension in the Business BASIC execution command line. Each /S file is an indirect command to the BASIC CLI, so each file must contain valid BASIC CLI commands and their arguments. Note that the CLI command !*filename* is analogous to the command SWAP "*filename*, so any program may be executed via STARTUP. Refer to *Business BASIC Reference Manual for Subroutines, Utilities, and BASIC CLI* for further information on the BASIC CLI utility.

For example, suppose the following files and their contents exist in the $SYS ($SY3) directory:

File	Contents
SHOWLOG.JB	`SLINE 0/L 80/W`

BACKUP.JB	`DIR DATA;MOVE/V/R DATABU`
	`DIR PGMS;MOVE/V/R PGMSBU`
	`DIR $SYS`

CLEARLOG.JB	`PRINT/H SYSLOG;DELETE SYSLOG` `CRAND SYSLOG`

You can execute the following command in the RDOS CLI:

```
BBASIC/S 8/J SHOWLOG/S BACKUP/S CLEARLOG/S
```

This command executes Business BASIC with eight jobs. STARTUP, invoked by the global /S switch, first passes SHOWLOG.JB to the CLI and attaches the job to the terminal on port 0 (SLINE command). The output will be shown on the terminal at

this port. Then, STARTUP passes BACKUP.JB to the CLI to back up the DATA and PGMS directories. Note that BACKUP.JB changes the current directory, and you must return to $SYS so that STARTUP can find and execute the remaining .JB file. Finally, STARTUP executes CLEARLOG.JB to print SYSLOG, delete it, and recreate it for future use. After this, STARTUP logs off the job, and the interpreter continues to bring up the system. If appropriate, the log-on banners are displayed.

Any program can be executed through the Business BASIC CLI from the .JB file. However, each time you use the Business BASIC CLI command START to begin a detached job to run your program, you must bring up a job to start the detached job. When deciding how many jobs you need to execute Business BASIC using automatic job execution, remember that STARTUP always requires one job. In addition, if any .JB file contains programs that start other jobs, a job must be executed for each job to be started. Additional jobs must also be executed if the user plans to log on to Business BASIC.

Executing Business BASIC with the /K global switch brings up Business BASIC with the kill flag set. When the kill flag is set, HELLO prevents nonprivileged users from logging on, and the Business BASIC spooler will not run. This allows any critical start-of-day processing done by the STARTUP (/S) facility to run without competition or concern for shared data. However, the same program in the STARTUP stream must clear the kill bit after critical processing is completed to begin normal usage. An example of this code follows:

```
0100 REM - Code to clear the Kill bit
0110 LET X=0
0120 STMB 0,9,X         :Get address of global switches
0130 STMB 10,0,X,32     :Clear the bit
```

It is sometimes useful to disable the log-on program during critical processing. To do this, set the kill bit to prevent nonprivileged users from logging on. An example of this code is shown below:

```
0100 REM - Code to set the Kill bit
0110 LET X=0
0120 STMB 0,9,X         :Get address of global switches
0130 STMB 10,1,X,32     :Set the bit
```

As indicated above, the Spooler shuts down automatically when the kill bit is set.

System and User Program Libraries

When Business BASIC is executed, it opens the system program library (BASIC.PL), if you have built one.

For more information, see "Program Library Builder" in the RDOS section of the "Utilities" chapter.

Spooler

The RDOS Business BASIC spooler enables multiple jobs to run that are generating print-image output to a limited resource, such as a line printer. In addition, programs can direct output to terminals or to terminal printing devices on Business BASIC or operating system multiplexor lines. This is accomplished by directing the output of the programs to special print-image files, called queuefiles, which are printed by a (typically detached) Spooler job. Batch jobs can also be executed through the Spooler mechanism.

For more information, see the RDOS section of the "Utilities" chapter.

Initialization Errors

At execution, special initialization errors can occur. These errors and possible explanations are shown below.

A Business BASIC system generated for a particular system configuration might not work on another configuration. Thus, a Business BASIC system generated for one configuration but run on another could generate a number of initialization errors.

```
Initialization error 1 Define user clock
```

There is no real-time clock generated into RDOS or a system error was returned by the .DUCLK system call.

```
Initialization error 3 Define INTEN, INTDS
```

A system error was returned by the system call .DEBL. This error should not occur. Please submit an STR (Software Trouble Report).

```
Initialization error 4 Open overlay file
```

A system error occurred while trying to access the Business BASIC overlay file (systemname.OL). Two possible errors are that the file could not be found and the file's use count is not zero.

```
Initialization error 5 Open (F)COM.CM
```

Either the (F)COM.CM file could not be found or the file is in use. Check to see if the other ground has the file opened or if the file's use count is not zero.

```
Initialization error 6 Read (F)COM.CM
```

A system error occurred while reading the (F)COM.CM file. This error should not occur. Please submit an STR.

Initialization error 9 Initialize system or library directory

$SYS or $LIB ($SY3 or $LIB3) has not been initialized, or an error occurred while attempting to initialize them. The directories could not be found in the current partition, or not enough subdirectories were generated into RDOS.

Initialization error 14 Open push file

A system error occurred while trying to open BASIC.PS. One possible error is a nonzero use count on the file. Use the RDOS CLI command CLEAR to clear the file's use count to zero.

Initialization error 15 Create BBASIC compiler task

A system error returned from .TASK while trying to create a user job. This error should not occur. Please submit an STR.

Initialization error 17 Create console task

A system error returned from .TASK while trying to create the input or output task for the master console. This error should not occur. Please submit an STR.

Initialization error 18 Create TTY1 task or IDEF interrupt handler

System error returned while trying to initialize the second TTY task. Possible causes for this error are a secondary console was generated into RDOS when it should not have been, not enough user defined devices were specified in the RDOS sysgen, or another program has already .IDEFed device codes 50 and 51.

Initialization error 21 Create ring detector task

System error returned while trying to create a modem ring detector task. This error should not occur. Please submit an STR.

Initialization error 22 IDEF multiplexor interrupt handler

A system error was returned while trying to initialize a multiplexor. Possible causes for this error are a multiplexor is generated into RDOS and Business BASIC with the same device code, not enough user-defined devices selected in RDOS, or another program has already .IDEFed this device.

`Initialization error 23 QTY:64 OPEN`

A system error was returned while Business BASIC was trying to open QTY:64. One possible cause for this error is that no multiplexor has been generated in RDOS.

`Initialization error 24 Create QTY monitor task`

A system error was returned while Business BASIC was trying to create the QTY monitor task. This error should not occur. Please submit an STR.

`Initialization error 25 Create TIMED INPUT task`

A system error was returned while Business BASIC was trying to create the timed input helper task. This error should not occur. Please submit an STR.

`Initialization error 27 Define window mapping`

System error occurred from the system calls .VMEM or .MAPDF. This error should not occur. Please submit an STR.

`Initialization error 28 Window space too small`

This error is caused by not having enough memory available in the logical address space to create a user program window as large as was specified with the local /M switch. If no /M was specified, then this error is caused by not being able to allocate the default program size of 24 Kbytes. To avoid this error, request fewer jobs, a smaller program size, or fewer resources in your Business BASIC sysgen. You can request fewer resources in your sysgen by limiting the number of channels, the number of lock areas, or the input and output buffer sizes.

`Initialization error 30 Insufficient extended memory`

There was not enough extended memory allocated to this ground to hold the requested number of jobs (/J) with the desired program size (/M or 24 Kbytes). To correct this error, increase the amount of memory with the RDOS CLI command SMEM, choose fewer jobs, or choose a smaller program size.

`No room for UFTs`

There are not enough channels specified in the RDOS system for the number of channels being requested by Business BASIC.

End of Chapter

Chapter 5
Security

This chapter describes the security features available in Business BASIC and how to use those features.

AOS Security

AOS provides extra security and file protection features for AOS Business BASIC. Normally, Business BASIC checks for the AOS assigned AA prefix on the username when privileged statements are entered, unless you generate your system so that all users have access to these statements. In general, only AA accounts may enter system calls (STMBs, STMCs, and STMEs). Regardless, the AOS operating system checks user profiles created by the PREDITOR utility and access control lists (ACLs) to maintain security. The user's ACL and profile prevent unauthorized access of files.

Logging On

HELLO is a Business BASIC program that handles logging on. A standard HELLO program comes in the $SYSLIB directory ($SYSLIB3 for triple precision systems).

At logon, the HELLO program checks the switches that were used when Business BASIC was executed and sets up the user status table. HELLO determines your terminal type and sets your terminal characteristics. HELLO also determines whether uppercase and lowercase input is allowed and whether eight-bit characters are allowed. If RLS2 is not found when Business BASIC is executed, the message `RLS2 NOT FOUND` is displayed, but if RLS2 is found, HELLO displays the pathname of RLS2.

The HELLO program can be modified to meet the needs of your installation.

AOS maintains a username and password security system. A son process bears the creator's username. Also, Business BASIC maintains a user status table associated with each job. Use this information to monitor and maintain the jobs on the system. (See the USERSTATUS file in the $DOC directory for information about the user status tables.)

Run-Only Programs

In AOS Business BASIC, run-only programs can be listed only by privileged users. This differs from run-only systems where no one can list programs, not even privileged users.

To make a program run-only for nonprivileged users, execute STMB 16,1. You can use the STMB statement in a program or execute it as a keyboard mode command. STMB 16,1 sets a run-only flag for Business BASIC program files. Execute STMB 16,1 to set this flag before doing a SAVE or REPLACE on the program. STMB 16,0 clears this flag. Only privileged users (AA accounts) can use this command.

Please note that run-only programs are not the same as generating Business BASIC to be a run-only system. Anyone with access to a run-only program can run it. However, generating Business BASIC to be a run-only system means that when Business BASIC is executed the prompt changes to `* RUN "` and users are only allowed to run existing programs. A run-only program refers to an individual program, whereas run-only systems affect all programs run on the system.

Directory, File, and Link Access

AOS allows you to restrict access to any file or directory by stipulating conditions for the file's use. You accomplish this by specifying an access control list (ACL) with the AOS CLI command ACL. The access control list contains a list of users who can access the file or directory, plus the types of access each user is entitled to.

The SUPERUSER privilege is an option for each user profile in AOS. SUPERUSER, if on, allows a user to change the ACL of any file or bypass all file access controls. Care should be used when allowing these privileges in user profiles.

With the AOS PERMANENCE command, you can protect files and directories from accidental deletion. A permanent file cannot be deleted unless you first specify PERMANENCE OFF.

You can also use the Business BASIC utility PROTECT to prevent users from listing a saved file. For more information see the AOS section of the "Utilities" chapter.

In terms of general access, AOS is a secure system. Only people who have user profiles can log on to the system, and only a privileged person can create user profiles. The default user profiles are nonprivileged, so the standard user cannot become privileged. AOS prevents any nonprivileged user from accessing any file unless the file's ACL allows this.

For more information refer to your operating system manuals.

RDOS Security

RDOS Business BASIC provides several security and file protection features not offered by the RDOS operating system. You can generate a run-only system by answering the appropriate question in the BSG dialog. You can also make Business BASIC programs run-only by using STMB 16. In addition, security features protect the system from outsiders, protect users from each other, and protect the integrity of your programs and data.

Logging On

After you execute Business BASIC, the message

```
DGC BBASIC x.xx
```

appears on the screen. Press the Esc key to start the log-on procedure.

If you run Business BASIC in the foreground and you are using operating system multiplexor support, you must type an uppercase or lowercase F (instead of pressing the Esc key) to log on to a foreground console. HELLO is a Business BASIC program that handles logging on. A standard HELLO program is supplied in the $SYS directory ($SY3 for triple precision systems), and only the system manager (an AA account) can modify it. This program uses a user account system, which is described later.

The HELLO log-on program maintains account IDs and password access to the system. Only valid user accounts can log on to the system. HELLO also places users in specific directories. Each user account can be allowed or denied access to certain system utilities and CLI commands. You can also cause a Business BASIC program (perhaps a menu program) to execute automatically when the account logs on. You can protect data by assigning file attributes that restrict the type of access and restrict access to a directory of files with a U$ERS file.

When you log on, HELLO determines your terminal type. If the program does not recognize your terminal or if the system is being used heavily, HELLO prompts you for the terminal type number. DASHER® display terminals are recognized as type 6.

After HELLO determines your terminal type, it uses items 18 and 19 of STMAs 3 and 4 to set the default primary and secondary unpend keys. The default unpend keys for type 6 terminals are Carriage Return <13> for the primary unpend key and New Line <10> for the secondary unpend key. After setting the defaults, HELLO reads the terminal's model ID to see what kind of Data General terminal is being used. If the terminal has a model ID, the primary unpend key is changed to New Line <10> and the secondary unpend key is changed to Carriage Return <13>. Primary and secondary unpend keys are not changed on terminals that do not have model IDs. Also, HELLO does not change the primary and secondary unpend keys if the terminal is the master console.

When you log on, HELLO prompts you for an account ID and a user ID. The

account ID is a two-letter code for the type of account: AA is a system manager, AB is a system programmer, OP is a system operator, and any other letters represent other account types the system manager wants. The user ID is a unique three-letter code for each user (usually the user's initials). HELLO then asks for a password (which is not echoed when entered) and checks it against the password in the user's account record. If they don't match, or if HELLO can't find the user ID in the user accounts file, HELLO logs the user off.

HELLO has a special system manager password (AA account) so you can set up your system and have access to HELLO, $SYS ($SY3), and the user account file. The system manager's password is DE2LA6. The user account file supplied with Business BASIC has passwords for the system operator (OP account) and the system programmer (AB account). The password for OP accounts is DEG2YQ, and the password for AB accounts is DE2ACE. You can change these passwords by modifying your HELLO program or your user account file.

Also, you can specify a directory for a user in the user's account file, and HELLO places the user in that directory. Otherwise, HELLO prompts you for a directory name. If the user responds with a directory that the user is not allowed into (perhaps restricted by a U$ERS file), HELLO logs the user off. Respond with a directory and a program, separated by a slash (/), to enter the directory and run the program (for example, "MYDIR/MYPROG"). You can disable logons during critical processing. Set the kill bit to prevent nonprivileged logons. An example is shown below:

```
0100 REM - Code to set the kill bit
0110 LET X=0
0120 STMB 0,9,X         :Get address of global switches
0130 STMB 10,1,X,32     :Set the bit
```

Run-Only Programs

Run-only programs can be listed only by privileged users. This differs from run-only systems where no one can list programs, not even privileged users.

To make a program run-only for nonprivileged users, execute STMB 16,1. You can use the STMB statement in a program or execute it as a keyboard mode command. STMB 16,1 sets a run-only flag for Business BASIC program files, that is, saved or replaced programs. Execute STMB 16,1 to set this flag before doing a save or replace on the program. STMB 16,0 clears this flag. Once you've set the run-only flag for the program, only an AA (system manager) account or privileged users (determined through use of the /Z switch at execution) can list it. Anyone with access to the file can run it.

Please note that run-only programs are not the same as generating Business BASIC to be a run-only system. Anyone with access to a run-only program can run it. However, generating Business BASIC to be a run-only system means that when Business BASIC is executed the prompt changes to `* RUN "` and users are allowed to run only existing programs. A run-only program refers to an individual program, whereas run-only systems affect all programs run on the system.

Menu Programs

You can specify a program to execute at logon, such as a master program that swaps to other programs. If you also set the forced SWAP flag, the program forces the user off the system when execution ends. This is called a menu program.

For example, the GM account has a menu program, GM.BB. When a GM user logs on, GM.BB executes and tells the user what programs he/she can execute. GM.BB then swaps to the program specified. This program executes, and control then returns to GM.BB, so that the user cannot list the program. GM.BB can also have ON IKEY, ON ERR, NEW, and BYE statements. ON IKEY captures attempts to interrupt the execution of the menu program; ON ERR captures runtime errors; NEW clears the working storage so that the user can't list the program; and BYE logs the user off. Figure 5-1 shows a menu program that displays the names of game programs. The user selects a game to execute or one of the commands STOP, ADD, or H (for help).

```
0010 REM Game menu program
0020 REM DATA FILE MENU.TX                    :MENU.TX contains names of programs.
0030 ON IKEY THEN GOTO 0340                   :If interrupted, go to IKEY handler.
0040 DIM IN$[80]
0050 PRINT "TYPE H FOR HELP"
0060 PRINT
0080 PRINT "GAMES CURRENTLY WORKING ARE"
0090 PRINT
0100 OPEN FILE[0,0],"MENU.TX"
0110 INPUT FILE[0],IN$                        :Get program names and comments.
0120 IF EOF(0) THEN GOTO 0150                 :End of program listings.
0130 PRINT IN$                                :Print names and comments.
0140 GOTO 0110
0150 REM End of program listings
0160 PRINT
0170 PRINT
0180 INPUT USING "","COMMAND: ",IN$           :Ask for program or command.
0185 STMA 14,IN$,0                            :Convert IN$ to uppercase.
0190 IF IN$="STOP" THEN GOTO 0380             :STOP command handler.
0220 IF IN$="ADD" THEN GOTO 0430              :ADD command handler.
0230 IF IN$[1,1]="H" THEN GOTO 0580
0250 ON ERR THEN GOTO 0290
0260 CLOSE
0270 SWAP IN$                                 :If not a command, SWAP to the program.
0280 GOTO 0010 :  Game menu program           :Go back to beginning when execution
0290 REM Invalid program name                 :returns.
0300 PRINT
0310 PRINT "ERROR INVALID COMMAND OR PROGRAM NAME"
0320 PRINT
```

UC-2003

Figure 5-1 Menu Program

```
0330 GOTO 0010 :  Game menu program
0340 REM IKEY handler
0350 CLOSE                                  :If interrupt key is hit, close the
0370 BYE                                    :file and BYE off the system.
0380 REM STOP command handler
0390 CLOSE                                  :if user types the STOP command,
0410 BYE                                    :close the file and BYE off the system.
0420 END
0430 REM ADD command handler
0440 INPUT USING "","PROGRAM NAME: ",IN$
0450 IF LEN(IN$)<20 THEN GOTO 0490 (>)
0460 PRINT
0470 PRINT "PROGRAM NAME TOO LONG MAX CHARACTERS IS 15"
0480 GOTO 0430 :  ADD command handler
0490 LET IN$[0]=FILL$(32)
0500 INPUT USING "","COMMENT: ",IN$[35,80]
0510 IF LEN(IN$)<80 THEN GOTO 0560 (>)
0520 PRINT
0530 PRINT "COMMENT FIELD TOO LONG"
0540 PRINT
0550 GOTO 0490
0560 PRINT FILE[0],IN$
0570 GOTO 0180
0580 REM HELP command handler
0590 PRINT
0600 PRINT "COMMANDS AVAILABLE ARE"
0610 PRINT "ADD -ADD TO THE PROGRAM LIST"
0620 PRINT "STOP -STOP"
0630 PRINT "PROGRAM-NAME -EXECUTE A PROGRAM"
0640 DELAY 20
0650 GOTO 0180
```

Figure 5-1 Menu Program *(concluded)*

If you generate a run-only Business BASIC system, create menu programs to display the program choices. You can also use a menu program to monitor all newly entered data or programs and to act as an interpreter. Menu programs can receive the data and start detached jobs that check for data integrity without disturbing the user's keyboard mode. Detached jobs can handle output queues, transaction logging, specialized file maintenance, etc. The default detach key is Ctrl-D for Business BASIC multiplexor support or Ctrl-C Ctrl-D for operating system multiplexor support. Use STMA 4,0 to redefine the detach key. For more information about detached jobs, see the ATTACH and START utilities in *Business BASIC Reference Manual for Subroutines, Utilities, and BASIC CLI.*

Directory, File, and Link Access

The Business BASIC *system directory* is the directory from which you executed Business BASIC. When Business BASIC is terminated, the system directory becomes the current directory in RDOS. The *user directory* is the directory you specified when you logged on to Business BASIC. Since each user can be in a different directory, you can have many user directories. However, Business BASIC uses only one system directory.

Users can change their user directories by executing the Business BASIC CLI command DIR. You can change the system directory using the Business BASIC CLI command SDIR. In addition, you can change the system directory and your user directory simultaneously by using the DIR statement or command. If you change the system directory during a Business BASIC session, the current RDOS directory is no longer the RDOS directory from which Business BASIC was executed. In addition, if you change the system directory while you are using the Business BASIC spooler, the spooler may be unable to find link files and your jobs may not be processed properly.

You can have a user log on in a specific directory by placing that directory in the user's account record in the ACCOUNTING file (see the section "Setting Up User Accounts"). If no directory is specified, the HELLO program prompts you for one.

To restrict a nonprivileged user to one directory, place that directory in the user's account record and do not set the flag for the Business BASIC CLI command DIR. This prevents a nonprivileged user from using the Business BASIC CLI command DIR.

The Business BASIC default is for only the AA account to use filenames with directory specifiers. Other accounts can use the following program to enable directory specifiers in file names:

```
0010  LET UST,WORD2=0
0020  STMB 1,1,14,UST          :Word address of user status table
0030  STMB 1,1,UST+1,WORD2     :Retrieve user status word 2
0040  LET WORD2=OR(WORD2,128):Set bit to allow directory specifiers
0050  STMB 2,1,UST+1,WORD2     :Copy word 2 back to memory
```

If you allow use of the BASIC CLI DIR command, you can restrict access to some directories by creating a U$ERS file in those directories. A U$ERS file is a text file that contains the account IDs of the users allowed in the directory. The BASIC CLI and the HELLO program check the U$ERS file in the directory a user requests. U$ERS entries take priority over ACCOUNTING file directory specifiers. Therefore, do not specify a directory for a user in the ACCOUNTING file unless the directory U$ERS file contains the user's account ID or the directory has no U$ERS file.

AA accounts (system managers) can automatically use the BASIC CLI DIR command and have access to all directories regardless of U$ERS files.

You can use the RDOS dash (-) and asterisk (*) conventions in the U$ERS entries.

Below is an example of a U$ERS file:

```
* !TYPE U$ERS
AB-
PRDGC-
**XYZ-
```

In this example, any user in the AB account group, the user PRDGC, and the user XYZ in any account group can use the current directory. You must include an asterisk (*) or a trailing dash (-) to match the terminal type number, which the system automatically appends to the account ID (or you can restrict certain terminal types from the directory).

The Business Basic DIR statement or command ignores U$ERS files. As a command, DIR returns an error message when entered by a nonprivileged user. As a statement, DIR can only be entered by AA users, but other users can run programs containing the statement.

Attributes

Unless you create the U$ERS file as a permanent attribute-protected file, anyone who has access to the directory can create a U$ERS file or modify an existing one.

You can restrict access to directories and files using the file attributes described with the CHATR command and the link attributes described with the CHLAT command. These are both Business BASIC CLI and RDOS CLI commands. A file's attributes permit or restrict reading, writing, renaming, deleting, or linking. A file's link attributes describe how the file can be accessed via a link.

Link entries allow access to any disk file or device through an alias. They allow users in different directories to use a single copy of a file. Link entries can point to other link entries up to ten levels. The final file in a set of links is called the resolution file. Deleting a link entry deletes the resolution file, if the resolution file's attributes allow that. To remove a link entry without deleting the resolution file, use UNLINK.

Link entries govern user access to reserved devices. If you establish a link to a magnetic tape or cassette device, initialize the device to activate the link. To use devices with names that do not conform to Business BASIC's reserved filename conventions, use links.

Usually during statements like OPEN, LOAD, SWAP, and CHAIN, the RDOS Business BASIC interpreter appends the filename to the user's current directory name. The directory name is stored in the User Status Table before the system call opens the file. The directory name in the User Status Table is changed when the BASIC CLI DIR command is executed. Thus, files are found in the user's directory, rather than in the RDOS system directory. If the file is not found, the interpreter tries the system call in the library directory, $LIB or $LIB3, so commands like RUN "CLI do not need links to $LIB:CLI in each user directory. Users can define directory specifiers when bit 8 of U.S2 in the User Status Table is set. Otherwise,

any filename containing a colon is rejected unless it is in the reserved filename table.

STMCs and reserved filenames are not appended to the user's current directory name. If no directory is specified by the user, no directory is specified by the interpreter. The operating system looks for the file in the current system directory, unless it is a file that is treated specially by the operating system. Since it is possible to change the current system directory at runtime (!SDIR, DIR, STMC 9), be careful about using links to reserved files and other special files that are called using STMCs.

For further information on devices, partitions, directories, files, links, and attributes, refer to your operating system manuals.

Setting Up User Accounts

The ACCOUNTING database in $SYS ($SY3) holds all passwords and accounts except for the AA (system manager) account. Only the AA account has access to $SYS ($SY3). The ACCOUNTING database determines the privileges for each user. These include the user's job priority, an optional log-on directory for the user, an optional BASIC program to execute at logon, and flags to allow the user some or all BASIC CLI commands.

The ACCOUNTING database is made up of these components:

File	Description
ACNTNG.DB	The physical file containing two logical files, ACNTIX and ACNTS.
ACNTNG.VL	The volume label file that contains information about the logical database file, ACNTNG.DB.
ACNTNG.TB	A table file that allows you to run FM on the ACCOUNTING database.
ACNTIX	A logical index file with a key representing the Account-Group/User-ID.
ACNTS	A logical data file that contains user accounts for logging on to Business BASIC.
ACCOUNTING	A link to ACNTNG.DB.

Use the File Maintenance (FM) utility to add users to the ACCOUNTING database. A table file, ACNTS.TB, for ACCOUNTING, is supplied. Respond ACNTS to the FM file prompt.

To change a field in the ACCOUNTING database, do the following:

- Press Carriage Return (New Line for DG/RDOS) to leave a field unchanged (or to not set it if this is the first time through the fields).
- Press the space bar to delete the entry in the field.
- Press any character to set the value in the field. When a field is set, it is displayed as an X.

The following example demonstrates adding a record to a GM account.

```
* RUN "FM
Data File Maintenance Rev. X.XX
FILENAME: ACNTS
BBASIC Accounts
 RECORD # 7_____ Account Group/User ID GM____ Password _____
Directory GAMES_____ Program GM.BB_____ Priority 19_ ADIR__
ACHATR_  Timeout X  No Messages_  FSWAP X  ACLI_  ASTART_  No NEWS_
Output queue _____
```

The meanings of the various fields are:

File	Description
Account Group/User ID	A two-letter code for the type of account and an optional three-letter code identifying the user. The AA account is reserved for use by the HELLO program. Accounts AB, OP, SP, SPOOL, P$RIN, and B$ATC already exist in the ACNTS file.
Password	An optional six-letter code to prohibit unauthorized users from logging on.
Directory	The log-on directory. If no directory is specified here, HELLO asks the user for a directory at logon.
Program	A program automatically executed at logon. If no program is specified, the user can request a program by responding "DIRNAME/PROGNAME" to HELLO's DIRECTORY prompt; otherwise, the user goes directly to keyboard mode.
Priority	A value from 13 to 255, default is 19. Priorities 0 to 12 are reserved for system tasks; assigning these priorities to a user results in the default priority.
ADIR	A character other than space allows the use of the BASIC CLI DIR command.
ACHATR	A character other than space allows the use of the BASIC CLI CHATR and CHLAT commands.

File	Description
Timeout	A character other than space automatically logs the user off after 10 minutes of inactivity.
No Messages	A character other than space prohibits sending messages (MSG, STMD 0) to this user.
FSWAP	A character other than space causes a SWAP to the program specified in the Program field. When the program terminates, control returns to HELLO, and HELLO logs the user off. A space causes HELLO to CHAIN to a specified program. When the program terminates, the user is in keyboard mode.
ACLI	A character other than space allows the use of the Business BASIC CLI program.
ASTART	A character other than space allows the use of the Business BASIC START program.
No NEWS	A character other than space prevents printing the NEWS file at logon.
Output queue	$LPT, unless otherwise specified.

The account GM was created without specifying a password; thus, a GM user does not need to enter one to log on. Program GM.BB, a menu program, will be swapped to by HELLO, keeping a GM user from going to keyboard mode. A GM user will be logged on at priority 19 and will receive any news and any messages. If inactive for 10 minutes, the user is logged off. Since no output queue is specified, the GM account's output goes to the default output queue, $LPT (the line printer).

The following figure illustrates the format of the ACCOUNTING database.

```
Data File: ACNTS
Record size: 128 bytes
File size: 200 records
Index file: ACNTIX
Key size: 6 bytes (5 used)


Field Description       Location        Size        Type

Status (value=1)           0              2        Integer
Account Group/User ID      2              5        String
(key in ACNTIX)
Reserved                   7              1
Password                   8              6        String
Directory                 14             20        String
Program                   34             14        String
Flags (status word)       48              2        Integer

* Bits for flags are numbered 15 - 0:

  15  FSWAP   (Program, if set above, will be SWAPped to instead
                of CHAINed to, forced SWAP).
  12  ASTART  (Allow use of the Business BASIC CLI command START,
                if bit 11 is set).
  11  ACLI    (Allow use of the Business BASIC CLI, will not run
                unless this is set).
  10  NONEWS  (Inhibit execution of the NEWS program).
  9   ACHATR  (Allow use of the Business BASIC CLI command CHATR,
                if bit 11 is set).
  8   ADIR    (Allow use of the DIR command, access to directories
                except $SYS or $SY3).
  6   NOMSG   (Inhibit receiving messages).
  5   TIMOUT  (Auto log-off after ten minutes of inactivity, time-out).

Reserved                  50              1
Priority                  51              1        Binary
Reserved                  52              2
Output queue (blank       54              6        String
for line printer)

*The rest of the file is used to store system logging information,
 if this feature is used.
```

UC-2004

Figure 5-2 ACCOUNTING Database Format

System Logging

RDOS Business BASIC provides a system logging facility. This facility monitors three types of system activity records produced by the HELLO program: log-on, log-off, and abort. The following programs enable you to use the system logging facility:

Program	Description
LOGINIT	Creates and initializes a log file named LOGGIN.
LOGDISP	Displays or prints log information from LOGGIN.
ACCOUNT	Updates the ACCOUNTING data file (described in Setting Up Accounts) with information from the LOGGIN file.
ACNTNGROLL	Selectively clears the resource usage fields in the ACCOUNTING database. Rolls current usage fields into total usage fields. Rolling involves the addition of current fields into total fields, followed by the clearing of all current fields.
ACNTNGRPT	Displays or prints a report using the data accumulated by the ACCOUNT program.

The LOGGIN File

LOGGIN, Business BASIC's account logging file, is a logical file located in the $SYS ($SY3) directory. The LOGGIN logical file is a circular file. This means that when the highest valid record is used, the logging routines that call LOGGIN automatically wrap around and overwrite the first record. This keeps the logging routines that use LOGGIN from producing errors due to a full LOGGIN.

LOGINIT

The LOGINIT program must be executed in the $SYS ($SY3) directory to enable system logging. None of the system logging utilities will work if this is not done. LOGINIT creates the LOGGIN physical and logical files and deletes any existing files named LOGGIN. You can use RUN, SWAP, or CHAIN "LOGINIT, or execute LOGINIT from the BASIC CLI.

Example

```
* RUN "LOGINIT
THIS PROGRAM CREATES AND INITIALIZES THE LOGGING FILE.
ARE YOU SURE YOU WANT TO DO THAT [NO]? YES
HOW MANY LOG RECORDS SHOULD BE ALLOWED [1999]?
```

LOGDISP

The LOGDISP program displays or prints a report showing the following information for each record in LOGGIN:

- The LOGGIN record number
- Account
- Operation (logon, logoff, or log abort)
- Date and time of each logon, logoff, or log abort
- Push level
- Job number
- Port number
- Directory
- CPU time
- Number of system (I/O) calls made
- Connect time

Following a display of LOGGIN, LOGDISP allows you to update the pointer in LOGGIN so that the records that have been displayed can be skipped in future uses of LOGDISP.

You can use RUN, SWAP, or CHAIN "LOGDISP. LOGDISP must be executed in the $SYS ($SY3) directory.

Example

At the prompts, Carriage Return was pressed to accept the defaults. The defaults are:

- Display at the terminal screen
- Display from the last update
- Show all logons, logoffs, and log aborts
- Display these records on the next run of LOGDISP.

To move the pointer in LOGGIN so only new records are displayed on next run of LOGDISP, enter Y at the prompt UPDATE LAST READ?.

```
* !LOGDISP
OUTPUT FILE [$TRO]?
DISPLAY ALL RECORDS [NO]?
SKIP LOGON RECORDS [NO]?

 REC  ACTION    ACCOUNT      DATE     TIME     P  J  L
1080 LOGOFF     SPOOL2   6/23/85    10:11:58   0  9 -1 $SPL   19.3  407 00:04
1079 LOGOFF     CLABC2   6/23/85    10:11:08   0  0 -1 CLASS   1.2  109 00:06
1078 LOGON      AAEYC2   6/23/85    10:08:45   0 11  3 $SYS/CLI
1077 LOGABORT   CLDGC2   6/23/85    10:07:57   0  9  3 CLASS
                INVALID ACCOUNT/PASSWORD AT 2070
1076 LOGON      CLABC2   6/23/85    10:05:05   0  0  3 CLASS


UPDATE LAST READ [NO]?
```

ACCOUNT

You use the ACCOUNT program in the $SYS ($SY3) directory to update the ACCOUNTING database with information from the LOGGIN file. ACCOUNT updates ACCOUNTING based on the log-on and log-off records in LOGGIN. The information contained in the LOGGIN records is listed in the description of the LOGDISP program. When you update the records, ACCOUNT displays or prints (depending on what you specify as the output file) a report of all the updated log-off records.

You can use RUN, SWAP, or CHAIN "ACCOUNT.

Example

```
* SWAP "ACCOUNT
LAST WRITTEN=1080    LAST ACCOUNTED=1066  # RECORDS=14   MAX RECORDS=1999
OUTPUT FILE [$TR0]:
 REC  ACTION    ACCOUNT      DATE     TIME     P  J  L
1080 LOGOFF     SPOOL2   6/23/85    10:11:58   0  9 -1 $SPL   19.3  407 00:04
1079 LOGOFF     CLABC2   6/23/85    10:11:08   0  0 -1 CLASS   1.2  109 00:06
```

ACNTNGROLL

The ACNTNGROLL program allows you to clear and/or roll the resource usage fields in the ACCOUNTING file. When you specify rolling, the current fields are added to the total fields and then the current fields are cleared. The resource usage fields from the ACCOUNTING database include:

- CPU time used
- Number of system (I/O) calls made
- Connect time

You can use RUN, SWAP, or CHAIN "ACNTNGROLL. You should use the ACNTNGRPT program both before and after using ACNTNGROLL. ACNTNGRPT uses the resource usage fields in the ACCOUNTING database to print a user activity report.

Example

```
* RUN "ACNTNGROLL
ROLL (1), CLEAR CURRENT (2), CLEAR TOTALS (3), CLEAR ALL (4): 1
```

ACNTNGRPT

The ACNTNGRPT program prints a 132-column report from the ACCOUNTING database using the ACCOUNT program data. This data shows current and accumulated values for CPU time used, system (I/O) calls made, and number of logoffs. Additionally, the date and time of last logoff and the number of logoffs for the period are shown. A new period starts each time the ACNTNGROLL program rolls the current values into total values.

You can use RUN, SWAP, or CHAIN "ACNTNGRPT.

Example

This example sends the report to the printer.

```
* RUN "ACNTNGRPT
OUTPUT FILE [$TR0]: $LPT
```

NEWS

The NEWS program prints a file on news items entered by the users. See NEWS in the RDOS section of the "Utilities" chapter.

End of Chapter

Chapter 6
Utilities

The Business BASIC utilities described in this chapter give you the ability to alter the characteristics of your files (for example, PROTECT or FIXFILE) or alter the way your system is operating (for example, the spooler or KILL).

The utilities are listed alphabetically beneath an operating system heading.

AOS Utilities

The AOS utilities are:

Utility	Function
DBFIX	Adjusts the characteristics of logical database files.
FIXFILE	Adjusts AOS file types.
LOCKS	Displays your current locks.
Program Library Builder	Builds system and user program libraries.
PROTECT	Protects Business BASIC program files.

The utilities are described on the following pages.

DBFIX *AOS Business BASIC CLI Command*

Adjusts the characteristics of logical database files.

Format

!DBFIX[global-switches] database

Arguments

database Name of the database that was copied.

Global Switches

/V Verify copied databases with a list of names on the generic file @OUTPUT.

What It Does

DBFIX converts the element size and adjusts the file type of logical database files (.VL and .DB) loaded from RDOS. If the links are moved from RDOS, DBFIX unlinks the existing links and creates new links. If links do not exist, DBFIX reports an unlink error and correctly links the files.

The element size is converted to a multiple of four for shared access by Business BASIC programs. The volume label (.VL) file is changed to AOS file type VLF and the database (.DB) file is changed to AOS file type DBF.

How To Use It

Execute DBFIX from the Business BASIC CLI. Enter the database name without the .DB extension. Templates are not permitted.

Example

The following example adjusts a database file named CUSTOMER that was loaded from RDOS.

```
!DBFIX/V CUSTOMER
Link fixed: CUST
Link fixed: CUSTI1
Link fixed: CUSTI2
Link fixed: CUSTI3
File fixed: CUSTOMER.DB
File fixed: CUSTOMER.VL
```

FIXFILE

AOS Business BASIC Command

Adjusts AOS file types.

Format

!FIXFILE filename type

Arguments

filename — Name of the file that was copied.

type — Numeric code or standard mnemonics representing AOS files.

What It Does

FIXFILE adjusts the AOS file type of filename to be the specified type. The type can be any of the numeric codes or the standard AOS mnemonics including three additional mnemonics. They are shown in the following table.

Code	Mnemonic	File Type
88	BBS	Business BASIC program (SAVEd)
89	VLF	Volume label file
90	DBF	Database file

FIXFILE also converts the element size to a multiple of four to allow shared access by Business BASIC programs. Use FIXFILE to change the default file type of UDF (which is assigned to imported RDOS Business BASIC programs) to the AOS file type BBS. DBFIX calls FIXFILE to convert logical database file types. Volume label file types (filenames with the .VL extension) are changed to VLF. Database files (filenames with the .DB extension) are changed to DBF.

FIXFILE does not check for INFOS statements. If FIXFILE is used with INFOS statements, it can create an erroneous save file.

How To Use It

Execute FIXFILE from the Business BASIC CLI.

Example

In the following example, the file type of DEMO is UDF. It is changed to BBS.

```
! FIXFILE DEMO BBS
```

LOCKS

AOS Business BASIC Utility

Displays your current locks.

Format

```
{ RUN   }
{ SWAP  } "LOCKS
{ CHAIN }
```

What It Does

LOCKS displays the locks you have set that have not been unlocked. Each job has a unique set of lock identifiers, so LOCKS' display includes each lock's filename and lock area. If an UNLOCK occurs while LOCKS is running, the lock associated with the UNLOCK disappears from the display.

How To Use It

To execute LOCKS, enter RUN, CHAIN, or SWAP "LOCKS.

When you use your terminal as a monitor to check the LOCKS that were set on another terminal, it does not matter which method of execution you use. However, to check the LOCKS that were set at the terminal where you are working, you must use either CHAIN or SWAP "LOCKS. This is true because when you use RUN "LOCKS or execute LOCKS from the BASIC CLI, all the files are closed. (RUN clears the common area, while the CLI always performs a CLOSE.) SWAP and CHAIN do not close the files, so you can check on the files' locks.

Program Library Builder

AOS Business BASIC Utility

Builds system and user program libraries.

Format

```
{ RUN   }
{ SWAP  }"PLB
{ CHAIN }
```

What It Does

Business BASIC supports program libraries, which are files containing many program SAVE file images, with a hashed index to the images. Normally, the system keeps the libraries open, so that a program can be executed without the overhead of an operating system open. Business BASIC supports user libraries as well as the system program library.

How To Use It

To execute PLB, enter RUN, CHAIN, or SWAP "PLB.

The system program library is the file BASIC.PL. This file is not supplied with Business BASIC; rather, the system manager must create it in $SYSLIB ($SYSLIB3) when Business BASIC is installed. The system manager must log on in the $SYSLIB ($SYSLIB3) directory and execute PLB to create a program library.

System Program Library

You can use the program library builder (PLB) to produce the system program library BASIC.PL using the program name list in LIB.CM. LIB.CM contains the default list of program names used to build the system program library. If the program you want to include in your program library is to be referenced by an alias name (such as a link or the lowercase version of the program name or link name), then you must also include all alias names on the same line in LIB.CM (or your input file) with the program name that they reference. You must separate the names by commas.

The frame size of 23 is adequate for the set of programs normally in the system program library and allows room to add user-developed programs. After building a program library you must terminate and re-execute Business BASIC to use the program library. Note that you will not be asked the frame size on subsequent runs of PLB.

Program Library Builder *Continued*

Example

While 700 blocks is adequate for the released modules, you need more space for many user modules. You also need a frame size greater than 23 for many user modules. Note that the frame size is a prime number that is 1 less than a multiple of 4.

```
* !CCONT BASIC.PL 700

* !PLB
PROGRAM LIBRARY FILE BUILDER - REV X.XX
LIBRARY NAME [BASIC.PL]?
INPUT FILE [$TRI]? LIB.CM
FRAME SIZE: 23
```

A verification listing of the modules is displayed and then the system returns to the Business BASIC prompt (*) for keyboard mode.

After you have built the system program library, you must terminate and re-execute Business BASIC to use the programs in the library.

You can execute any of the programs in the system program library via CHAIN, SWAP, or RUN simply by prefixing the name with a "#". Thus to execute the CLI from the library rather than from $SYSLIB ($SYSLIB3), you can use any of the following commands:

```
* RUN "#CLI"
```
or
```
* CHAIN "#CLI"
```
or
```
* SWAP "#CLI"
```

User Program Library

You can also use the program library builder to build user program libraries. User libraries are built in exactly the same way as BASIC.PL. (User libraries are usually smaller.) If the program you want to include in your program library is to be referenced by an alias name (such as a link or the lowercase version of the program name or link name), then you must also include all alias names on the same line in LIB.CM (or your input file) with the program name that they reference. You must separate the names by commas.

To maximize performance, you should create the library contiguously. You must select a prime number as the frame size (minimum size of 3), large enough to contain all of the program and link names. Optimum sizes are prime numbers that are 1 less than a multiple of 4 such as 3, 7, 11, or 19. This maximizes the utilization of sectors that are assigned to the library index.

Program Library Builder *Continued*

Note that although there can be 8 entries per frame, the nature of hashing will probably prevent you from achieving that density, so estimate a frame size 50 to 75 percent larger than you require.

The frame size depends on the number and names, including aliases, of programs to be included in the program library. No firm rules for selecting a frame size can be given. A rule of thumb is to select a prime number that is 1 less than a multiple of 4 and greater than one quarter of the number of programs. For instance, the BASIC.PL system program library contains approximately eighty programs (including the programs in LIB.CM), and a frame size of 23 suffices.

Place the program names to be included in the library in a test file that will be entered at the INPUT FILE [$TRI]? prompt. If the initial frame size is too small, delete the library and execute PLB again selecting a larger frame size.

Example

1) In the example user library build below, TMPLIB (a random file) has a frame size of seven. Note that the programs use aliases (i.e., links and case-sensitive names). The program ADVENTURE has an alias link name of ADVENT. Accepting the default name ends the PLB session.

```
* RUN "PLB
PROGRAM LIBRARY FILE BUILDER - REV X.XX
LIBRARY NAME [BASIC.PL]? TMPLIB
INPUT FILE [$TRI]:?
CREATING NEW LIBRARY
FRAME SIZE: 7

NAME(S): WUMPUS,wumpus
PROGRAM ADDED:  WUMPUS
LINK ADDED: wumpus
NAME(S): ADVENTURE,ADVENT,adventure,advent
PROGRAM ADDED: ADVENTURE
LINK ADDED: ADVENT
LINK ADDED: adventure
LINK ADDED: advent
NAME(S): TREK,trek
PROGRAM ADDED: TREK
LINK ADDED: trek
NAME(S): TREK.OL,trek.ol
PROGRAM ADDED: TREK.OL
LINK ADDED: trek.ol
```


Program Library Builder *Continued*

```
NAME(S): CUBIC,cubic
PROGRAM ADDED: CUBIC
LINK ADDED: cubic
NAME(S):

72 BLOCKS IN USE.
```

By building TMPLIB as a random file, you can determine the size needed for the contiguous library. Link names (additional names for a program in a library) follow the program name and a comma on entry. You can use the Business BASIC CLI command XFER to copy and reload any program library, including BASIC.PL, while the system is running. However, make sure that you do not overwrite the library while users are reading from it. If you change the frame size, the hashing algorithm cannot find entries for users who have already opened the library. Thus for BASIC.PL, Business BASIC must be shut down to change the frame size.

2) In the example below, the library is made contiguous for speed and extra size. An allowance for growth is made too.

```
!LIST TMPLIB
TMPLIB                36864 UDF 01/25/85 13:51 01/25/85 [003725] 0


!CCONT MYLIB.PL 120
!XFER TMPLIB MYLIB.PL/N
36864 BYTES TRANSFERED
!DELETE TMPLIB
```

To use a user program library, open it for reading. Use STMA 20 to pass the channel number to Business BASIC. This can be done in a program or as statements similar to the following:

```
0010 OPEN FILE (1,4),"MYLIB.PL" :Open the library.
0020 STMA 20,1                  :Indicate that it is a user library.
0030 SWAP "%ADVENTURE"          :Now use it.
```

Just as the "#" character is used to indicate a program from the system program library, the "%" character is used to indicate a program from the user program library.

After you issue STMA 20,1, you can use channel 1 in a subsequent OPEN FILE statement.

PROTECT

AOS Business BASIC Utility

Protects Business BASIC save files (programs).

Format

```
{RUN  }
{SWAP } "PROTECT
{CHAIN}
```

What It Does

PROTECT modifies a SAVE file in such a way that the user is prohibited from LISTing the file.

How To Use It

To execute PROTECT, enter RUN, CHAIN, or SWAP "PROTECT. The utility prompts `FILE TO BE 'PROTECTED':`. Respond with the name of a program that has been saved. Next, the utility prompts `DO YOU WANT TO CHANGE LINE NUMBERS? (Y or N)`. Respond N if the program is ever invoked by another program with the statement SWAP filename THEN GOTO linenumber or CHAIN filename THEN GOTO linenumber.

To exit PROTECT, press New Line when prompted for the filename.

Example

This example shows PROG1 being protected and the line numbers being changed.

```
* RUN "PROTECT

FILE TO BE 'PROTECTED': PROG1
DO YOU WANT TO CHANGE LINE NUMBERS? (Y or N): Y

YOUR FILE IS NOW PROTECTED.


FILE TO BE 'PROTECTED':
```

RDOS Utilities

The RDOS Business BASIC utilities are:

Utility	Function
ANALYZE	Displays the system status of an RDOS Business BASIC system.
KILL	Ends Business BASIC.
LOCKS	Displays your current locks.
NEWS	Log-on message program.
OPCLI	OPCLI utility commands display and set job priorities, interrupt active jobs, send messages, and force users off the system.
Program Library Builder	Builds system and user program libraries.
PROTECT	Protects Business BASIC program files.
QUICKILL	Terminates Business BASIC without warning messages.
Spooler	Enables running multiple jobs generating print-image output. The Spooler includes the FORMSCHG, FORMSCLI, QTYPE, and SPCLI utilities.
VACUUM	Creates macros to work with subdirectories and partitions.

The utilities are described on the following pages.

ANALYZE — *RDOS Business BASIC Utility*

Displays the status of an RDOS Business BASIC system.

Format

```
{RUN  }
{SWAP }"ANALYZE
{CHAIN}
```

What It Does

ANALYZE provides information on the status of a Business BASIC system at the time a BREAK.SV file is created by the RDOS interrupt character sequence. Use ANALYZE to investigate why a system is hung or to analyze a currently running system. The utility displays three reports on the system status.

How To Use It

To execute ANALYZE, enter RUN, CHAIN, or SWAP "ANALYZE.

The utility prompts you for the name of the file to be analyzed and the name of an output file to receive the analysis. If you press CR for output filename, the system sends the output to your terminal; otherwise, it creates the output file you name. To ANALYZE the current system, press CR instead of supplying an input filename.

The first part of the report displays the line table.

The second part of the report displays the User Status Table (for more information see appendix A, the User Status Table).

The third part of the report displays a task control block (TCB) for every task in the system. These include TCBs for the Business BASIC jobs listed in part two. Each TCB is preceded by its address. Business BASIC uses words 0-11, 15, and 16 of the TCB, with the following meanings:

WORD	MNEMONIC	CONTENTS
0	TPC	User PC (B0-14) and carry (B15)
1	TAC0	AC0
2	TAC1	AC1

ANALYZE *Continued*

WORD	MNEMONIC	CONTENTS
3	TAC2	AC2
4	TAC3	AC3
5	TPRST	Status bits and priority
6	TSYS	System call word
7	TLNK	Link word to next TCB
10	TUSP	For a Business BASIC job, pointer to User Status Table
11	TELN	Overlay number while task is mapped out
12	—	Not used
13	—	Not used
14	—	Not used
15	TSP	Stack pointer while task is mapped out
16	TFP	Frame pointer while task is mapped out

Since printouts are set up for 132-column paper, modify ANALYZE if you use 80-column paper. Change line 20 of the ANALYZE utility as follows:

For 132-column paper: `0020 DATA 40,16,16,132,0,0`
For 80-column paper: `0020 DATA 40,16,16,80,0,0`

Example

In the first part of this example, Port 0 (23200 is the address of the User Status Table) represents the job running at the master console. Port 1 (100000) represents the secondary console, which was not generated in the BASIC system, but for which an address was assigned anyway. Ports 2 through 5 are multiplexor lines and no jobs are attached to them. For more information on the second part of this report, see the USERSTATUS file in the $DOC directory. For information about the third part of this report, refer to the TCB table above.

ANALYZE *Continued*

```
* RUN "ANALYZE
FILENAME:
OUTPUT FILE:
L I N E   T A B L E
 23200
100000
     0
     0
     0
     0

(000016) =  23200
(000414) =    445

J O B   N U M B E R  0
U.S     020040
 SGNO
 NSWP
U.S2    015602
 ASTR
 ACLI
 ACHA
 ADIR
 AFFN
 ALOW
U.S3    015602

U.CPU      176
U.IOU       85
U.LOG      946
U.TCB   000445
U.JBN        0
U.TLN        0
U.DS   $SYS
U.ACN  AASCW6
U.PNM  ANALYZE
U.PSL        0
U.P     032000
U.PL      2608
U.D     071301
U.DL       319
U.DA    000000
U.CMX    16384
U.PRI       19
U.SPR  000 000
U.ESC   014215
```

ANALYZE *Continued*

```
U.PR          2
U.DFO   $LPT
U.DEL   231 177
U.CAN   004 030
U.DLS   134 015
U.PND   012 015
U.ESD   377 033
U.TBN    000217
U.TBC   23356 1
U.TBS   23346 0
U.AC0    000000
U.MLN       132
U.TPW        80
U.TTS        14
U.TCC        80
U.PAD   <0><0>
U.PCN    000000
U.TYP         6
U.ERC         0
U.USR    000000


         000000
U.SCD    177777
U.STN      9060
U.RAD    000000
I.CCN         0
I.SZE        40
I.INP   23471 1
I.OUT   23471 1
I.BGN   23456 0
I.END   23502 0
O.CCN         0
O.SZE        80
O.INP   23546 1
O.OUT   23546 1
O.BGN   23502 0
O.END   23552 0
U.ACT    000000
U.SVA    073000
U.TRS    072000
U.STP     30272
U.EDA         0
U.EPA         0
```

ANALYZE *Continued*

```
U.LFS        0
U.LCH   000000
U.DPN  012 015
U.REC   000000
U.SCH   177777
U.UCH  377 377
CHN1   001 005
CHN2   377 377
CHN3   377 377
CHN4   377 377
CHN5   377 377
CHN6   377 377
CHN7   377 377
CHN8   377 377
CHN9   377 377
CHN10  377 377
CHN11  377 377
CHN12  377 377
CHN13  377 377
CHN14  377 377
CHN15  377 377

T A S K   C O N T R O L   B L O C K
TPC     6647 1
TAC0    046714
TAC1    000021
TAC2    000405
TAC3    023200
TPRST   000023
TSYS    017077
TLNK    177777
TUSP    023200
TELN    177777
TID     000000
TTMP    000000
TKLAD   000000
TSP     072010
TFP     000000
```

```
================================================================================
          .
          .                :The User Status Table for each job is printed
          .
          .
================================================================================
```

ANALYZE *Continued*

```
ADDR    000424
TPC    17052 0
TAC0    017030
TAC1    000001
TAC2    000000
TAC3    000000
TPRST   120000
TSYS    017030
TLNK    000613
TUSP    000000
TELN    000000
TID     000000
TTMP    000000
TKLAD   000000
TSP     072003
TFP     000000

          .
          .              :The Task Control Block for every
          .              :job in the system is printed

ADDR    000445
TPC     6647 1
TAC0    046714
TAC1    000021
TAC2    000405
TAC3    023200
TPRST   000023
TSYS    017077
TLNK    177777
TUSP    023200
TELN    177777
TID     000000
TTMP    000000
TKLAD   000000
TSP     072001
TFP     000000
```

KILL

RDOS Business BASIC Utility

Terminates Business BASIC and returns control to RDOS.

Format

```
{RUN  }
{SWAP } "KILL
{CHAIN}
```

What It Does

KILL terminates the Business BASIC system and returns control to RDOS. It resides in the $SYS (or $SY3) directory.

How To Use It

To execute KILL, enter RUN, CHAIN, or SWAP "KILL. KILL prompts you for the AA (system manager) account password if you are not logged on with an AA account.

If there are no other users on the system, KILL executes QUICKILL. If other users are logged on, KILL displays the status of all users currently using the system. KILL asks you for the number of minutes it should wait before killing the system and for an appropriate message to request all users to log off. If you specify 0 minutes for KILL to wait, KILL assumes a 30-second wait. KILL forces the message to all users at half the time interval since the last message. Note that you can interrupt (IKEY) the KILL program to stop shutdown procedures for any reason. When the time is up, KILL shuts down Business BASIC, returning control to the RDOS master console, whether or not users are still logged on to Business BASIC. See Figure 6-1 for an example of KILL.

KILL *Continued*

```
* RUN "KILL
```

The screen is cleared.

```
01 I C R USER            EDIT            ABFPG 02
02 I C C MYDIR           SCRATCH         ABALB 03
MINUTES ALLOWED TO LOG OFF: 1
MESSAGE: GO HOME
```

The screen is cleared again and the following is displayed at the console where KILL was executed.

```
01 I C R USER            EDIT            ABFPG 02
02 I C C MYDIR           SCRATCH         ABALB 03
```

At terminals 2 and 3, the following messages would appear:

```
SYSTEM WILL SHUT DOWN IN 0:56.      PLEASE LOG OFF.
GO HOME

SYSTEM WILL SHUT DOWN IN 0:36.      PLEASE LOG OFF.
GO HOME

SYSTEM WILL SHUT DOWN IN 0:15.      PLEASE LOG OFF.
GO HOME

SYSTEM WILL SHUT DOWN IN 0:05.      PLEASE LOG OFF.
GO HOME

SYSTEM IS SHUTTING DOWN.
GO HOME
```

After this final message, jobs 1 and 2 are logged off. At the terminal where KILL is run, the following message appears:

```
BUSINESS BASIC IS DOWN   DG/RDOS STILL RUNNING
```

UC-2005

Figure 6-1 KILLing a Business BASIC System

LOCKS

RDOS Business BASIC Utility

Displays your current locks.

Format

```
{RUN  }
{SWAP } "LOCKS
{CHAIN}
```

What It Does

LOCKS displays the locks you have set that have not been unlocked. Each job has a unique set of lock identifiers, so LOCKS' display includes each lock's filename and lock area. If an UNLOCK occurs while LOCKS is running, the lock associated with the UNLOCK disappears from the display.

How To Use It

To execute LOCKS, enter RUN, CHAIN, or SWAP "LOCKS.

When you use your terminal as a monitor to check the LOCKS that were set on another terminal, it does not matter which method of execution you use. However, to check the LOCKS that were set at the terminal where you are working, you must use either CHAIN or SWAP "LOCKS. This is true because when you use RUN "LOCKS or execute LOCKS from the BASIC CLI, all the files are closed. (RUN clears the common area, while the CLI always performs a CLOSE.) SWAP and CHAIN do not close the files, so you can check on the files' locks.

NEWS

RDOS Business BASIC Utility

Log-on message program.

Format

```
{RUN  }
{SWAP } "NEWS
{CHAIN}
```

What It Does

RDOS Business BASIC has an automatic log-on message program called NEWS in the $SYS ($SY3) directory. The HELLO program executes NEWS during a log-on session unless the user's account record in ACCOUNTING has a flag set to prohibit NEWS.

How To Use It

To execute NEWS, enter `RUN, CHAIN, or SWAP "NEWS`.

NEWS maintains two files: NEWS.LG and NEWS.DS. NEWS.LG is an index file maintained by NEWS; its key entries are user accounts. Each key's pointer is the date that user account last received the news. NEWS.DS is a text file you must create using a text editor. It contains the text of your news. NEWS.DS should reside in the $LIB ($LIB3) directory. NEWS expects the most recent news to be at the beginning of the NEWS.DS file, with the older news following it. Each news item can be several lines long, but you must start each news item with a line beginning with a backslash (\) and the date as follows:

```
\DATE mm/dd/yy
```

where mm/dd/yy are the month, day, and year; and mm begins at the seventh character position.

NEWS scans the NEWS.DS file for a line beginning with \DATE mm/dd/yy. The system compares this date with the date found in NEWS.LG for that user account. If the news in NEWS.DS is newer than the date in NEWS.LG, the user can choose to read or skip it. You can create log-on messages in HELLO to handle this choice; for example, `DO YOU WANT THE NEWS AS OF 06/23/85?` If the news in NEWS.DS is older than the date in NEWS.LG for that user account, then HELLO assumes the user has seen the news and does not display it. NEWS checks dates only when HELLO executes it. You can use RUN "NEWS at any time to read all of the news. HELLO does not execute NEWS for detached jobs.

OPCLI

RDOS Business BASIC Utility

Performs system operator functions.

Format

```
{RUN  }
{SWAP } "OPCLI
{CHAIN}
```

What It Does

The OPCLI performs system operator functions. OPCLI commands display and set job priorities, interrupt active jobs, send messages, and force users off the system.

The following OPCLI commands can be used fully by any user:

Command	Meaning
ALL	Broadcast a message.
CLI	Simulate the RDOS CLI environment.
IKEY	Interrupt execution of a job.
KILL	Terminate Business BASIC and return control to RDOS.
LOCKS	Display current locks.
MSG	Send a message to a terminal.
POP	Exit from the OPCLI and clear the common area.
SHOW	Display the current state of a job.
STAT	Display the status of all jobs.

The following OPCLI command can be used by any user for display purposes only. It can be used fully only by AA and OP accounts:

Command	Meaning
PRI	Display or set job priorities.

OPCLI *Continued*

The following OPCLI commands can be used only by AA and OP accounts:

Command	Meaning
BYE	Forces a job to log off.
FALL	Forces a message to all terminals.
FMSG	Force a message to a single terminal.

How To Use It

To execute OPCLI, enter RUN, CHAIN, or SWAP " OPCLI.

The OPCLI prompt is an angle bracket (>).

The OPCLI commands follow in alphabetical order.

Example

The following example executes the OPCLI.

```
* RUN "OPCLI

>
```

ALL

OPCLI Command

Broadcast a message.

Format

ALL[global-switches] "message"

Arguments

message — The message you want to broadcast.

Global Switches

/A — Broadcasts a message to all terminals including those not logged on to Business BASIC.

What It Does

ALL broadcasts the message to all terminals. The message does not appear if a job attached to a terminal has a no-message flag set (FALL can force a message to these terminals). The optional /A switch sends the message to all terminals generated into the system, whether they are logged on to Business BASIC or not.

How To Use It

Execute ALL by entering it from the OPCLI. Enclose a message in quotes.

Example

This message appears at all terminals logged on.

```
>ALL/A "DO YOUR WEEKLY REPORTS"
FROM 0:AAXXX DO YOUR WEEKLY REPORTS
>
```

BYE

OPCLI Command

Forces jobs to log off.

Format

BYE [job-number] [job-number...]

Arguments

job-number A job number for any user other than an AA (system manager) user.

What It Does

BYE forces any job to log off. Only those logged on with AA or OP accounts can use this command.

How To Use It

Use the OPCLI command STAT to get the job number. Then enter BYE followed by the job numbers you want to log off. If you specify no job, BYE logs you off the system.

Example

This command logs job number 16 off the system.

```
>BYE 16
```

CLI

OPCLI Command

Swaps to the CLI utility.

Format

```
CLI
```

What It Does

This causes you to SWAP to the Business BASIC CLI utility so that you can perform CLI commands without having to exit this program and then run CLI. Enter POP while in CLI and you will be returned to the OPCLI.

How To Use It

Execute CLI by entering it at the OPCLI prompt (>). Refer to *Business BASIC Reference Manual for Subroutines, Utilities, and BASIC CLI* for an explanation of this command.

Example

```
>CLI
!POP
>
```

FALL *OPCLI Command*

Forces a message to all terminals.

Format

FALL[global-switches] "message"

Arguments

message The message you want to force.

Global Switches

/A Forces a message to all terminals generated into the Business BASIC system.

What It Does

FALL broadcasts the message to all Business BASIC terminals that are logged on. The message appears even if a job attached to a terminal has a no-message flag set. The optional /A switch sends the message to all terminals generated into the system, whether they are logged on to Business BASIC or not.

Only users logged on with AA or OP accounts can use this command.

How To Use It

Execute FALL by entering it from the OPCLI. Enclose the message in quotes.

Example

This message goes to all terminals on the system.

```
> FALL/A "SYSTEM MAINTENANCE BEGINS AT 10:00 AM TODAY"
```

The following is displayed at all terminals.

```
FROM 0:AAAAA SYSTEM MAINTENANCE BEGINS AT 10:00 AM TODAY
```

FMSG *OPCLI Command*

Forces a message to a terminal.

Format

FMSG[global-switches] port-number "message"

Arguments

port-number	The port number of the terminal to receive the message.
message	The message you want to force.

Global Switches

/W	Wait 30 seconds for a reply.

What It Does

FMSG forces a message to a terminal whether or not a job attached to the terminal has a no-message flag set. The MSG command honors the no-message flag.

How To Use It

The optional /W switch suspends processing for 30 seconds to wait for a reply from the terminal.

If you are using operating system multiplexor support, you cannot use the /W switch to wait for a reply to a terminal that is not logged on to Business BASIC. However, if the terminal is logged on, you can use the /W switch.

Only those logged on with AA or OP accounts can use this command.

Use the OPCLI command STAT to find the port number of the terminal. Then enter FMSG from the OPCLI followed by the terminal number and a message enclosed in quotes.

Example

At port 0, the following is entered and a message is received.

```
>FMSG/W 2 "ARE YOU THE SYSTEM MANAGER WHILE BOB IS ON VACATION"
LINE 2 :AAAAA6  NO
>
```

The following is displayed at port 2 and NO is entered by the person using this terminal.

```
FROM 0:AAGGG  ARE YOU THE SYSTEM MANAGER WHILE BOB IS ON VACATION? NO
```

IKEY *OPCLI Command*

Interrupts a job's execution.

Format

IKEY job-number ...

Arguments

job-number A job number.

What It Does

Use IKEY to stop jobs caught in infinite loops.

IKEY interrupts a job whether the job has set an IKEY disable flag or an ON IKEY trap.

How To Use It

Use the OPCLI command STAT to find the job number.

Execute IKEY by entering it at the OPCLI prompt (>).

Example

This command interupts job 16.

```
>IKEY 16
```

KILL *OPCLI Command*

Terminates Business BASIC and returns control to RDOS.

Format

```
KILL
```

What It Does

KILL swaps you to the Business Basic KILL utility. It terminates the Business BASIC system and returns control to RDOS.

How To Use It

Execute KILL by entering it at the OPCLI prompt (>). Refer to the KILL utility described earlier in this chapter for more information.

KILL asks for the AA (system manager) account password if you are not logged on with an AA account.

LOCKS

OPCLI Command

Displays your current locks.

Format

```
LOCKS
```

What It Does

LOCKS displays the locks you have set that have not been unlocked. Each job has a unique set of lock identifiers, so LOCKS' display includes each lock's filename and lock area. If an UNLOCK occurs while LOCKS is running, the lock associated with the UNLOCK disappears from the display.

LOCKS displays all locks that have not been unlocked along with each lock's filename and file area. An UNLOCK of a lock during the display removes the lock's values from the display.

How To Use It

Execute LOCKS by entering it at the OPCLI prompt (>).

For more information, refer to the RDOS Business BASIC LOCKS utility described earlier in this chapter.

MSG *OPCLI Command*

Sends a message to a terminal.

Format

MSG[global-switches] port-number "message"

Arguments

port-number	The port number of the terminal to receive the message.
message	The message you want to send.

Global Switches

/W	Wait 30 seconds for a reply.

What It Does

MSG sends the message to one terminal. The message does not appear if a job attached to the terminal has a no-message flag set. (FMSG forces the message even if the flag is set.) The optional /W switch suspends processing for 30 seconds to wait for a reply from the terminal. If you are using operating system multiplexor support, you cannot use the /W switch to wait for a reply from a terminal that is not logged on to Business BASIC. However, if the terminal is logged on, you can use the /W switch.

How To Use It

Execute MSG by entering it at the OPCLI prompt (>).

Use the OPCLI command STAT to find the port number of the terminal.

The message must be enclosed in quotes.

Example

At port 0, the following is entered and a message is received.

```
>MSG/W 2 "HOW ABOUT LUNCH"
LINE 2 :AAAAA6  YES!
>
```

The following is displayed at port 2 and YES! is entered by the person using this terminal.

```
FROM 0:AAGGG  HOW ABOUT LUNCH? YES!
```


POP *OPCLI Command*

Exits from the OPCLI and clears the common area.

Format

```
POP
```

What It Does

POP terminates Business BASIC's OPCLI program and returns you to your previous level. POP clears the common area, which is used to pass arguments.

How To Use It

Execute POP by entering it at the OPCLI prompt (>).

Example

This example shows a POP to return from the OPCLI.

```
* RUN "OPCLI
>POP
*
```

PRI

OPCLI Command

Displays or sets job priorities.

Format

PRI job-number [priority]

Arguments

job-number A job number.

priority A priority value.

What It Does

PRI displays a job's priority (if there is no priority argument), or it modifies a job's priority.

How To Use It

Assignable priorities range from 13 to 255. (Priorities from 0 to 12 are reserved for system tasks.) Jobs with low number priority values receive higher priority from the system scheduler. An attempt to assign a priority in the range 0-12 results in the default priority, 19, being assigned.

Use the OPCLI command STAT to display the job numbers and the priorities. Then enter PRI from the OPCLI.

Example

This sets job 16's priority to 13.

```
>PRI 16 13
```

SHOW *OPCLI Command*

Displays the current state of a job.

Format

SHOW[global-switches] job-number ...

Arguments

job-number A job number.

Global Switches

/R Displays job status repeatedly.

What It Does

SHOW displays the current state of a job.

How To Use It

Execute SHOW by entering it at the OPCLI prompt (>).

If you include the optional /R global switch, SHOW displays a job's status repeatedly until you press the interrupt key. If you do not include the /R switch, you can display the status of several jobs.

Use the OPCLI command STAT to display the job numbers.

Example

Job 1 is running a program, with priority 13, in directory USER. The program executing is FM; the number of the statement executing is 9614. The account is ABBBB6. The terminal being used is terminal 1. There is one nested SWAP. The program size is 10022 bytes. The CPU usage is 19.4 seconds, and the input/output usage (number of system calls) is 1271. The account logged on at 13:59. The job's user status word 1 is 020102 and status word 2 is 001425 (octal). The account is allowed to change directories and use CHATR, the BASIC CLI, and START.

```
>SHOW 1

JOB=1 RUN XQT PRI=13 DIR=USER PROG=FM/9614 ACCT=ABBBB6 TERM=1 SWAP=1
SIZE=10022 CPU=19.4 I/O=1271 LOGON=13:59 020102 001425 ADIR ACHATR ACLI
ASTART
```

STAT *OPCLI Command*

Runs the Business BASIC STAT utility.

Format

```
STAT
```

What It Does

STAT displays the active job numbers, their I/O status, their locations, and their run status when STAT is executed without switches.

How To Use It

Execute STAT by entering it at the OPCLI prompt (>).

Refer to *Business BASIC Reference Manual for Subroutines, Utilities, and BASIC CLI* for more documentation on the STAT utility and its arguments and switches.

Program Library Builder *RDOS Business BASIC Utility*

Builds system and user program libraries.

Format

```
{RUN  }
{SWAP }"PLB
{CHAIN}
```

What It Does

Business BASIC supports program libraries, which are files containing many program SAVE file images, with a hashed index to the images. Normally, the system keeps the libraries open, so that a program can be executed without the overhead of an operating system open. Business BASIC supports user libraries as well as the system program library.

If the program you want to include in your program library is to be referenced by an alias name (such as a link or the lowercase version of the program name or link name), then you must also include all alias names on the same line in LIB.CM (or your input file) with the program name that they reference. You must separate the names by commas.

How To Use It

To execute PLB, enter RUN, CHAIN, or SWAP "PLB.

The system program library is the file BASIC.PL. This file is not supplied with Business BASIC; rather, the system manager must build it in $SYS ($SY3) after Business BASIC is installed. The system manager must log on in the $SYS ($SY3) directory and execute PLB to create a program library.

System Program Library

You can use the program library builder (PLB) to produce the system program library BASIC.PL using the program name list in LIB.CM. LIB.CM contains the default list of program names used to build the system program library. The frame size of 23 is adequate for the set of programs normally in the system program library and allows room to add user-developed programs. After building a program library, you must terminate and re-execute Business BASIC to use the program library. Note that you will not be asked the frame size on subsequent runs of PLB.

Program Library Builder *Continued*

Example

While 700 blocks is adequate for the released modules, you need more space for many user modules. You also need a frame size greater than 23 for many user modules. Note that the frame size is a prime number that is 1 less than a multiple of 4.

```
* !CCONT BASIC.PL 700

* !PLB
PROGRAM LIBRARY FILE BUILDER - REV X.XX
LIBRARY NAME [BASIC.PL]?
INPUT FILE [$TRI]? LIB.CM
FRAME SIZE: 23
```

A verification listing of the modules appears and then the system returns to the Business BASIC prompt (*) or keyboard mode.

After you have built the program library, you must shut down and re-execute Business BASIC to use the programs in the library.

You can execute any of the programs in the system program library via CHAIN, SWAP, or RUN simply by prefixing the name with a "#." Thus to execute the CLI from the library rather than from $LIB ($LIB3), you can use any of the following commands:

```
* RUN "#CLI
```
or
```
* CHAIN "#CLI
```
or
```
* SWAP "#CLI
```

User Program Libraries

You can also use the program library builder to build user program libraries. User libraries are built in exactly the same way as BASIC.PL. (User libraries are usually smaller.) If the program you want to include in your program library is to be referenced by an alias name (such as a link or the lowercase version of the program name or link name), then you must also include all alias names on the same line in LIB.CM (or your input file) with the program name that they reference. You must separate the names by commas.

To maximize performance, you should create the library contiguously. You must select a prime number as the frame size (minimum size of 3), and the number must be large enough to contain all of the program and link names.

Program Library Builder *Continued*

Optimum sizes are prime numbers that are 1 less than a multiple of 4 such as 3, 7, 11, or 19. This maximizes the utilization of sectors that are assigned to the library index. Note that although there can be 8 entries per frame, the nature of hashing will probably prevent you from achieving that density; so estimate a frame size 50 to 75 percent larger than you require.

The needed frame size depends on the number and names, including aliases, of programs to be included in the program library. No firm rules for selecting a frame size can be given. A rule of thumb is to select a prime number that is 1 less than a multiple of 4 and greater than one quarter of the number of programs. For instance, the BASIC.PL system program library contains approximately 80 programs (including the programs in LIB.CM), and a frame size of 23 suffices.

Place the program names to be included in the library in a text file that will be entered at the INPUT FILE [$TRI]? prompt. If the initial frame size is too small, delete the library and execute PLB again selecting a larger frame size.

Example

In the example user library build below, TMPLIB (a random file) has a frame size of seven. Note that the programs use aliases (i.e., links and case-sensitive names). The program ADVENTURE has an alias link name of ADVENT. Accepting the default name ends the PLB session.

```
* RUN "PLB
PROGRAM LIBRARY FILE BUILDER - REV X.XX
LIBRARY NAME [BASIC.PL]? TMPLIB
INPUT FILE [$TRI]:?
CREATING NEW LIBRARY
FRAME SIZE: 7

NAME(S): WUMPUS,wumpus
PROGRAM ADDED:  WUMPUS
LINK ADDED: wumpus
NAME(S): ADVENTURE,ADVENT,adventure,advent
PROGRAM ADDED: ADVENTURE
LINK ADDED: ADVENT
LINK ADDED: adventure
LINK ADDED: advent
NAME(S): TREK, trek
PROGRAM ADDED: TREK
LINK ADDED: trek
NAME(S): TREK.OL,trek.ol
PROGRAM ADDED: TREK.OL
LINK ADDED: trek.ol
```


Program Library Builder *Continued*

```
NAME(S): CUBIC,cubic
PROGRAM ADDED: CUBIC
LINK ADDED: cubic
NAME(S):

72 BLOCKS IN USE.
```

By allowing TMPLIB to be built as a random file, you can easily determine the needed size for the contiguous library. Links, additional names for a program in a library, are indicated by names separated by commas following the program name on entry.

You can use the BASIC CLI command XFER to copy and reload any program library, including BASIC.PL, while the system is running. However, make sure that you do not overwrite the library while users are actually reading from it and that you do not change the frame size. If you change the frame size, the hashing algorithm cannot find entries for users who have already opened the library. Thus for BASIC.PL, Business BASIC must be shut down to change the frame size.

In the example below, the library is made contiguous for speed and extra size. An allowance for growth is made too.

```
!LIST TMPLIB
TMPLIB              36864 C       01/25/85 13:51 01/25/85 [003725] 0


!CCONT MYLIB.PL 120
!XFER TMPLIB MYLIB.PL/N
36864 BYTES TRANSFERED
!DELETE TMPLIB
```

For you to use a user program library, it must be opened for reading. Use STMA 20 to pass the channel number to Business BASIC. This can be done in a program or as commands similar to the following:

```
0010 OPEN FILE (1,4),"MYLIB.PL" :Open the library.
0020 STMA 20,1                  :Indicate that it is a user library.
0300 SWAP "%ADVENTURE"          :Now use it.
```

Just as the # character is used to indicate a program from the system program library, the % character is used to indicate a program from the user program library.

After you issue STMA 20,1, you can use channel 1 in a subsequent OPEN FILE statement.

PROTECT

RDOS Business BASIC Utility

Protects Business BASIC save files (programs).

Format

```
{RUN  }
{SWAP } "PROTECT
{CHAIN}
```

What it Does

PROTECT modifies a SAVE file in such a way that the user is prohibited from listing the file.

How To Use It

The PROTECT and PROTECT.OL programs are in the $SYS and $SY3 directories. To access them, create links to them from the directory that contains the programs to be protected. If you want all users to be able to access the PROTECT utility, create the links in the $LIB and $LIB3 directories.

To execute PROTECT, enter RUN, CHAIN, or SWAP "PROTECT. The utility prompts `FILE TO BE 'PROTECTED':`. Respond with the name of a program which has been saved. Next, the program prompts `DO YOU WANT TO CHANGE LINE NUMBERS? (Y or N)`. If the program you are protecting will ever be invoked by another program using the statement SWAP filename THEN GOTO linenumber or CHAIN filename THEN GOTO linenumber, you must respond N to this prompt.

To exit PROTECT, press New Line when prompted for the filename.

Example

This example shows PROG1 being protected and with the line number changed:

```
* RUN "PROTECT

FILE TO BE 'PROTECTED': PROG1
DO YOU WANT  TO  CHANGE  LINE NUMBERS? (Y or N) Y

YOUR FILE IS NOW PROTECTED.


FILE TO BE 'PROTECTED':
```

QUICKILL

RDOS Business BASIC Utility

Terminates Business BASIC without warning messages.

Format

```
{RUN  }
{SWAP } "QUICKILL
{CHAIN}
```

What It Does

QUICKILL terminates the Business BASIC system without warnings and returns control to RDOS. It resides in the $SYS (or $SY3) directory.

How To Use It

To execute QUICKILL, enter RUN, CHAIN, or SWAP "QUICKILL.

You must have access to $SYS ($SY3) to use QUICKILL. QUICKILL asks for the AA (system manager) account password before you can shut down the system. QUICKILL doesn't warn the users or give a time limit. Only use QUICKILL in emergencies or in Business BASIC systems with only one job. Otherwise, use the KILL utility.

RDOS Business BASIC Spooler

The RDOS Business BASIC spooler enables the running of multiple jobs that are generating print-image output to a limited resource, for example, a line printer. In addition, programs can direct output to terminals or to terminal printing devices on Business BASIC or operating system multiplexor lines by directing the output of the job to special print-image files, called queue files, that will be printed by a spooler job. The Business BASIC spooler programs are in the $SPL ($SPL3) directories.

Starting the Spooler

After loading the spooling system modules, use one of the following methods to start spooler jobs.

1) Log on as the SP account to automatically execute the SPCLI utility. The SP account is in the ACCOUNTING file supplied with Business BASIC.

   or

   Log on in the $SPL ($SPL3) directory and run the SPCLI to bring up spooler jobs as an AA user. The prompt for the SPCLI is the pound sign (#).

   ```
   * RUN "SPCLI
   #
   ```

   After you have performed one of the above procedures, enter the SPCLI command START at the prompt. This command executes the spooler as a detached job. If more than one device should be supported for simultaneous output of queue files, then additional START commands can be given to start as many spoolers as needed, or the SPCLI PMAX facility described below can be used to allow a spooler to start additional jobs. The number of spooler jobs running simultaneously will be limited based on the available jobs.

2) Log on as the SPOOL account in the ACCOUNTING file. It automatically starts the spooler. The SPOOL account is in the ACCOUNTING file supplied with Business BASIC. This account enables a nonprivileged user to start the spooler. However, logging on a terminal with the SPOOL account renders that terminal nonfunctional. Once the user logs on, the spooler is started and the terminal cannot be used. Enter a Ctrl-D to detach the spooler.

3) By using the SPOOL account and the STARTUP program, you can easily start one or more spoolers as part of the routine of bringing up

RDOS Business BASIC Spooler *Continued*

Business BASIC each day. To do this, create a file called SPOOLER.JB in $SYS ($SY3) that contains the following command for STARTUP: `START SPOOL/A`

When you bring up Business BASIC, the command line can use the /S switch to automatically start the spooler as a detached job. Thus, the following command brings up Business BASIC with 10 jobs, 2 of which are running spoolers:

```
BASIC/S 10/J SPOOLER/S/2
```

Shutting Down the Spooler

The spooler automatically terminates when you run the KILL program to bring down Business BASIC. Each spooler job periodically looks at the Kill bit, which is set by the KILL program; if the bit is set, the spooler job will stop looking for queue files to output and will log itself off. If a SPOOLER is in the process of outputting a queue file when KILL is run, sufficient time should be allowed for the output to complete; otherwise, KILL can abort the SPOOLER with a queue file in use.

Another way to terminate a spooler is with the SPCLI command SUSPEND. By placing the /K global switch on the SUSPEND command, you can terminate a specific spooler by using its job number as an argument to SUSPEND. If no argument is given, the lowest-numbered spooler job aborts.

Queue Files, Queue Names, Queue Tables, and Devices

A queue file is a unique user spool file located in the $SPL ($SPL3) directory. Queue files are created by prefixing a "?" to a queue name and opening the file in mode 1 or 2. Business BASIC will then create a queue file on disk with a name conforming to the convention queuename$nnnn, where nnnn is a unique numeric identifier appended by the interpreter.

A queue name is the logical name for a system output queue that can be accessed by multiple users concurrently. The queue name consists of from one to five alphanumeric characters; an example might be LPT, which would be linked to the device $LPT and accessed in user programs by using OPEN ?LPT. Links from queue names to devices are listed in a table called the queue table, which can be displayed with the SPCLI DEV command. The queue table for the background is QTABLE0 and the queue table for the foreground is QTABLE1. The SPCLI PROC command can be used to access the other ground's queue table. Links are entered into the queue table with the SPCLI LINK command, removed with UNLINK, and changed with RELINK.

RDOS Business BASIC Spooler *Continued*

If Business BASIC is running in both grounds, the spooler can be used in one of two ways:

1) SPOOLER jobs can be running in both grounds with separate queue tables. Queue names or device names referred to by QTABLE0 cannot be referred to by QTABLE1.

2) SPOOLER jobs can be running in only one ground. You can use queue names and device names defined in that ground's queue table to print queue files from both grounds.

The following creates a queue file where the queue name is LPT.

```
0100 OPEN FILE[15,2],"?LPT"
```

This statement opens a queue file LPT$nnnn. All queue files are created in the $SPL ($SPL3) directory. The automatic generation of unique queue file names allows any number of jobs to refer to the same queue name concurrently. This also allows any job to OPEN the same queue name on different channels at the same time.

In the same pass through a data file, a single program can generate a detailed report, a summary report, and an exception report. Queue names can be established for any output device or terminal supported by Business BASIC and RDOS. Queue name assignments are made by using LINK to create links to devices with the following conventions:

Convention	Device
$PRn	Printer with form feed on a Business BASIC multiplexor line
$TRn	Printer without form feed on a Business BASIC multiplexor line
$LPT	Line printer
filename	Other RDOS file/printer

In these conventions, n is the Business BASIC port number of the printer generated in Business BASIC.

One or more copies of the spooler programs are run as detached jobs and scan the $SPL ($SPL3) directory for completed queue files. Queue files that are being created or have already been output to their device are skipped. When the spooler finds a completed queue file, it determines the queue file's destination device. In order for the queue file to be output, the device and the queue file must meet certain criteria: the device must not be in use, the device

RDOS Business BASIC Spooler *Continued*

must have the same form name as specified in the queue file, and the queue file's priority must be within the range currently set for the device. If all criteria are met, the spooler outputs the queue file; otherwise, the spooler checks to see if the queue name is linked to other devices. Thus, if a given queue name is linked to multiple devices, the first available device that has the proper forms and priority settings will be used.

If a given queue name is linked to multiple devices, then multiple line printers and printing terminals can be used interchangeably. By running multiple spoolers or setting the SPCLI command PMAX greater than zero, you can output multiple queue files simultaneously to different devices. Each spooler job can only output one queue file at a time, but a spooler will start another PRINTER job to output the queue file under certain conditions. In this case, the spooler job will be free to look for other queue files and perhaps start other PRINTER jobs. The spooler will start a PRINTER job for each queue file to be output as long as the current number of PRINTER jobs logged on to the system is less than the value of PMAX; otherwise, the spooler job itself will output the queue file. (See PMAX.)

Setting Up a Control Record

The spooler deletes a queue file once it has been output. It is possible to direct the spooler to retain queue files for some specified amount of time and to output multiple copies of a given queue file. For you to do this, a queue file should contain a control record as the first record of the file with the following conventions. There can only be one control record per queue file.

Byte 1 must contain the value 1 (not the ASCII character 1). Refer to the example below.

The remainder of the record includes any combination of the following fields separated by a comma:

R=r

r is the number of days the queue file is to be retained. The default is 0.

U=u

u is the date, in the form mmddyy, until which the queue file is to be retained. The default is today's date.

RDOS Business BASIC Spooler *Continued*

5C=c

c is the number of copies to be printed. The default is 1.

P=p

p is the priority level of the queue file. Valid priorities are 0 - 255. The default is 255.

F=form-name

form-name is the user-defined name of the form on which this queue file should be printed. The default form-name is DEFAULT.

If the control record is not set up correctly, the user gets an error in the error summary file when the spooler tries to output the queue file. Use the SPCLI LOG command to list the contents of the error summary file.

The following example shows the code to print three copies of a file and retain the file for seven days.

```
0050 OPEN FILE[15,2],"?LPT"
0060 PRINT FILE[15],"<1>R=7,C=3"
```

Overstriking Text

The Business BASIC Spooler also allows you to overstrike text (for example, to print in boldface or use underscores). To overstrike, make sure that byte 1 of the PRINT FILE statement contains the code <21> followed by any text that is to be printed. The Spooler prints the text but does not generate a line feed. Instead, the printhead is returned to the left margin, and the Spooler prints the next PRINT FILE statement on the same line. The Spooler returns to the left margin of the same line every time it encounters the code <21> in the first byte of the PRINT FILE statement.

The following example shows the code to print boldface text and to underscore this text. Below the code example is an example of the printed output at the device.

```
0050 OPEN FILE (15,2),"?LPT"
0060 PRINT FILE(15),
0070 PRINT FILE(15),
0080 FOR I=1 TO 4
0090   PRINT FILE(15),"<21>Company Name" :Prints Company Name in bold
0100   PRINT FILE(15),"<21>___________" :and underscored.
```


RDOS Business BASIC Spooler *Continued*

```
0110 NEXT I
0120 PRINT FILE(15),                    :Skips to the next line (because
                                        :<21> is not in this code line).
0130 PRINT FILE(15),              :Skips a line between heading and text.

0140 PRINT FILE(15),"Address line 1"
0145 PRINT FILE(15),"Address line 2"
0150 PRINT FILE(15),"City, State"
0160 CLOSE FILE(15)
0170 END
```

The printed output for this program is:

```
Company Name

Address line 1
Address line 2
Address line 3
```

Forms and Priorities

When a spooler finds a queue file and a destination device, it checks whether the device's current forms and priority settings allow the output of the queue file. If a queue file must be printed on special forms or if the need for the printout is particularly high, the queue file will have a control record written by a user at the beginning of the queue file. For example, the queue file that results in the output of a department's paychecks can have a control record that was created with the following line:

```
0200 PRINT FILE[15],"<1>P=10,F=CHECKS"
```

To make sure that the checks are printed right away, a user might first set the high and low priorities of the destination device so that only queue files of priorities 0 through 10 would be output. Then, if the current forms on the device were not CHECKS, the user would redefine the forms and mount the checks on the device.

After the queue file had been output, the user could remount other forms and redefine the priority range to resume standard operation.

The priority of a particular queue file is defined in the control record; if no priority is specified, 255 is assumed. The current priority range for a

RDOS Business BASIC Spooler *Continued*

particular device is redefined using the SPCLI command PRIORITY. Queue files with priorities outside of the current range are not output to the device until the device's priority range is redefined.

Forms are defined using the FORMSCLI program. Several different types of forms can be defined for each device. If no particular types of forms are specified in a queue file, that queue file will be output when the destination device is set to its DEFAULT forms.

When the type of form mounted on the device is to be changed, use the command FORMS to change the form name to a pre-defined non-existent form name; mount the desired forms on the device; and use the command FORMS to change the form name to the name of the form you mounted.

The non-existent form name used for changing forms will never be used in a control record, but it must be added to the FORMS file. Thus, when a device is assigned this form name, the spooler ignores queue files that will be printed on the device. The non-existent form name used in the examples in this manual is SUSPEND.

Batch Jobs

Batch jobs can be executed through the spooler mechanism. When the spooler finds a queue file and notices that the queue name was linked to a device with the /B switch, the queue file will not actually be output to the device. Instead, it is used as a command file by a BATCH job. The same criteria that must be met by a standard queue file also apply to batch queue files: the device must be free, forms must be defined, and so forth. Typically, you should set the batch queue's forms to DEFAULT and its priority range to 0 through 255; thus, control records need not be used in the batch queue file.

For example, if BMAX is 1 and no BATCH jobs are currently logged on, the spooler starts a new job that runs the BATCH program (and if no jobs are available, the spooler waits until a job logs off). If BMAX is 0, the spooler swaps to BATCH. However, the BATCH program starts a slave job that will execute the Business BASIC commands contained in the batch queue file. Thus, at least one job must be available for a batch queue file to be executed successfully; if the BMAX mechanism is used, two jobs are needed.

The following is an example of a program that could be run by a user to create a batch job that dumps certain disk files to tape. This example could be used as a model for a daily backup procedure. The example assumes that $LPT and MT0: exist as devices and as Business BASIC reserved files and that a batch queue XXX has been defined, perhaps with the SPCLI command LINK XXX $LPT/B.

RDOS Business BASIC Spooler *Continued*

```
0010 OPEN FILE[0,1],"?XXX"       : Open a batch queue file.
                                 : This queue file will not be output to a
                                 : device; rather, it will be used as a
                                 : command file by a BATCH job because
                                 : of the /B on the SPCLI LINK command.
0020 PRINT FILE[0],"AAAAA,DE2LA6,DZ0"  : The first line of the file contains
                                 : the answers to HELLO's questions. The
                                 : batch job that will execute this batch
                                 : queue file supplies these answers to the
                                 : slave job it will start.  The following
                                 : several lines are Business BASIC
                                 : commands that the batch job will force
                                 : to the slave job.
0030 PRINT FILE[0],"STMA 10,1,<34>?LPT<34>" : The slave job will define its
                                 : default output device to be ?LPT.  Thus
                                 : the /L switches on the CLI commands to
                                 : follow will lead Business BASIC to
                                 : create queue files containing the
                                 : listings of files dumped during the
                                 : backup. These queue files will later be
                                 : output to $LPT through the spooler
                                 : mechanism.
0040 PRINT FILE[0],"!INIT MT0;DUMP/L MT0:0 -.SR;DIR WORK" : One mag tape file
                                 : and one /L queue file are created when
                                 : the .SR files in DZ0 (where the slave
                                 : job logged on) are dumped.
0050 PRINT FILE[0],"!DUMP/L MT0:1 -.LS"    : Another magnetic tape file and
                                 : another /L file are created.
0060 PRINT FILE[0],"!RELEASE MT0"
0070 PRINT FILE[0],"BYE"  : The slave job will log itself off when it's done.
0080 CLOSE                       : Closing this batch queue file makes it
                                 : fair game for the spooler. Depending on
                                 : the value of BMAX, the spooler will
                                 : either start a job to run BATCH or it
                                 : will swap to BATCH itself.  BATCH will
                                 : then read the batch queue file and force
                                 : the commands to another job it will
                                 : start. After the entire batch queue file
                                 : is processed, the BATCH job will either
                                 : log itself off or pop back to spooler.
0090 PRINT "Batch queue file created; backup will start momentarily."
                                 : This message is printed to the terminal
                                 : of the user who ran this program.  The
                                 : user could then run the PED utility
                                 : to watch the started jobs in action.
0100 END
```


RDOS Business BASIC Spooler *Continued*

Summary

The following procedure sets up $LPT as a spooler device. Modify the procedure to set up other devices.

1) Generate the RDOS system with $LPT. You should also specify $LPT as a Business BASIC reserved file.

2) Check your directories for any disk files named $LPT. You must delete or unlink any $LPT files; then release and reinitialize the directory.

3) Bring up Business BASIC with one detached spool job.

4) Move to $SPL, run FORMSCLI and create the FORMS file. This is required once for the spooler.

5) In $SPL, run the SPCLI, and link LPT to $LPT.

6) In $SPL, run FORMSCHG and enter FORMS $LPT DEFAULT.

7) Refer to the device file in your program as "?LPT"; for example,
```
0100 OPEN FILE[13,2],"?LPT"
```

Use the DIR command with care. Since it changes the directory in which Business BASIC is running, it is possible to lose access to the link running the printer. Links to devices must be used when the operating system device name does not match the Business BASIC reserved filename conventions of $*** and ***: (e.g., $LPT and MT0:).

Spooler Utilities

The spooler utilities are:

Utility	Function
FORMSCHG	Allows the user to assign the forms for output devices.
FORMSCLI	Maintains the user-definable FORMS file.
QTYPE	Displays a queue file.
SPCLI	Maintains the table of queue names and device assignments.

FORMSCHG

Spooler Utility

Allows the user to assign the forms for output devices.

Format

```
{RUN  }
{SWAP } "FORMSCHG
{CHAIN}
```

What It Does

The FORMSCHG program is the control program and user interface for all spooler output. It allows the user to change the forms in the output device, display the allowable form names and descriptions, and display the linked output devices.

Each command and its parameters are discussed below.

The FORMSCHG commands are:

Command	Action
BYE	Logs you off the system.
DEV	Lists the current device assignments.
DISPLAY	Displays the contents of the FORMS file.
FORMS	Changes the current FORMS assignment.
POP	Exits from FORMSCHG and clears the common area.
QUIT	Exits from FORMSCHG and retains the common area.

How To Use It

To execute FORMSCHG, enter RUN, CHAIN, or SWAP "FORMSCHG.

When you run FORMSCHG, the prompt is the percent sign (%).

BYE *FORMSCHG Command*

Logs you off the system.

Format

BYE

What It Does

BYE terminates the FORMSCHG program and logs you off Business BASIC.

How To Use It

Execute BYE by entering it at the FORMSCHG prompt (%).

Example

This command logs you off Business BASIC.

```
% BYE
```

DEV *FORMSCHG Command*

Lists the current device assignments.

Format

DEV[global-switches]

Global Switches

/L List to the default output queue. Use the Business BASIC CLI command GQUE to retrieve the default output queue.

What It Does

DEV displays the current state of the queue table, including queue names, device names, form names, priorities, and status switches. The status field is established by the SPCLI command LINK. Note that the status field is filled with asterisks (*) if more than five switches are set for the queue name; however, the status in the queue table is unaffected.

How To Use It

Execute DEV by entering it at the FORMSCHG prompt (%).

Example

Four assignments are made: LPT to the line printer loaded with the form APCHECK and printing all queue file priorities; LP2 to a printer with form feed on port 2 with no forms loaded and all priorities; TR to a printer without form feed on port 12 with form NP513-2 and priorities from 0 (high) to 127 (low) and with queue files to be retained for one day after printing; and LPX to a printer with form feed on port 4 with the default form DEFAULT loaded and all priorities. The status field is established by the SPCLI LINK command.

```
% DEV

 QTABLE0                06/23/85          13:41:47
RECORD    QNAME    DEV    STATUS      PRIORITY        FORMS
                                      HI      LO

   1        LPT    $LPT                 0     255      APCHECK
   2        LP2    $PR2                 0     255      SUSPEND
   3        TR     $TR12        S       0     127      NP513-2
   4        LPX    $PR4                 0     255      DEFAULT
```


DISPLAY *FORMSCHG Command*

Displays the contents of the FORMS file.

Format

DISPLAY

What It Does

DISPLAY shows the user the contents of the FORMS file.

How To Use It

Execute DISPLAY by entering it at the FORMSCHG prompt (%).

Example

This example displays the contents of a sample FORMS file.

```
% DISPLAY

FORM NAME     D E S C R I P T I O N                USAGE

9110PP-1      ONE PART 9 1/2 X 11 UNRULED 20#        0
9513P-2       TWO PART 9 1/2 X 11 RULED              0
DEFAULT       DEFAULT FORM N261411D                  0
N261411D-1    ONE PART 14 7/8 X 11                   0
N61011-1      ONE PART 10 5/8 X 11                   0
SUSPEND       DUMMY ENTRY TO CHANGE FORMS            0
```

FORMS *FORMSCHG Command*

Changes the current FORMS assignment.

Format

FORMS device-name form-name

Arguments

device-name Any output device supported by Business BASIC.

form-name The name of a form existing in the FORMS file that you want to assign to device-name.

What It Does

This command changes the form-name currently assigned to the output device.

How To Use It

Execute FORMS by entering it at the FORMSCHG prompt (%).

While changing forms, assign the device a non-existent form that has been defined in the FORMS file for that purpose. This prevents the spooler from starting the device while the new forms are being mounted.

Example

This changes the form assignment for the line printer, $LPT, to the form-name INVOICE, if the form has been set up in the forms control file FORMS. This command will change the forms entry for all queue names linked to the device.

```
%FORMS $LPT INVOICE
```

POP *FORMSCHG Command*

Exits from FORMSCHG and clears the common area.

Format

POP

What It Does

POP terminates Business BASIC's FORMSCHG program and returns you to your previous level. POP clears the common area, which is used to pass arguments.

How To Use It

Execute POP by entering it at the FORMSCHG prompt (%).

Example

POP returns the user to the FORMSCHG utility.

```
* RUN "FORMSCHG
% POP
*
```

QUIT *FORMSCHG Command*

Exits FORMSCHG and retains the common area.

Format

QUIT

What It Does

QUIT terminates the FORMSCHG program retaining the common area and returns you to your previous level.

How To Use It

Execute QUIT by entering it at the FORMSCHG prompt (%).

Example

QUIT returns you to keyboard mode.

```
* RUN "FORMSCHG
% QUIT
*
```

FORMSCLI

Spooler Utility

Maintains the user-defined FORMS file.

Format

```
{RUN  }
{SWAP } "FORMSCLI
{CHAIN}
```

What It Does

The FORMSCLI is the control program that maintains the user-definable FORMS file. Among other things, it allows the user to build and change the forms definitions within the file.

The FORMSCLI commands are:

Command	Action
ADD	Adds an entry to the forms file.
BYE	Logs you off the system.
CHANGE	Changes a form name or description.
CLI	SWAPs to the CLI program.
CREATE	Creates the forms files.
DELETE	Deletes entries from the FORMS file.
DISPLAY	Displays the contents of the FORMS file.
FORMSCHG	Swaps to the FORMSCHG program.
POP	Terminates the FORMSCLI and clears the common area.
QUIT	Terminates the FORMSCLI and retains the common area.
SPCLI	Swaps to the SPCLI program.

How To Use It

To execute FORMSCLI, enter RUN, CHAIN, or SWAP "FORMSCLI.

When you run the FORMSCLI, the prompt is the dollar sign ($).

ADD *FORMSCLI Command*

Adds an entry to the forms file.

Format

ADD form-name 'form-description'

Arguments

form-name	The name of a form to be added to the FORMS file. Any character other than single quotation mark (') or blank is allowed.
form-description	A string enclosed in single quotes that describes the form.

What It Does

ADD enters form-names and their form-descriptions to the FORMS file.

How To Use It

Execute ADD by entering it at the FORMSCLI prompt ($).

Example

```
$ ADD SUSPEND 'DUMMY ENTRY TO CHANGE FORMS'
```

Set the form SUSPEND. It is used to cause the spooler to ignore queue files that will be printed on the device; thus, it allows the operator to change the form on the device. It will not be used in a control record.

```
$ ADD N9110PP-1 'ONE PART 9 1/2 X 11 UNRULED 20#'
```

Add the form N9110PP-1 and its description to the FORMS file.

BYE *FORMSCLI Command*

Logs you off the system.

Format

BYE

What It Does

BYE terminates the FORMSCLI program and logs you off Business BASIC.

How To Use It

Execute BYE by entering it at the FORMSCLI prompt ($).

Example

```
$ BYE
```

CHANGE *FORMSCLI Command*

Changes a form name or description.

Format

CHANGE form-name new-name

CHANGE/D form-name 'new-form-description'

Arguments

form-name	The name of a form to be changed. It already exists in the FORMS file. Any character other than single quote (') or blank is allowed.
new-name	The new form name to be added to the FORMS file. Any character other than single quote (') or blank is allowed.
new-form-description	A string enclosed in single quotes that describes the form.

Global Switches

/D	Change the form's description rather than its name.

What It Does

CHANGE renames forms or rewrites their descriptions in the FORMS file.

How To Use It

Execute CHANGE by entering it at the FORMSCLI prompt ($).

Example

This changes the form named 9110J-1 to 9110P-1.

```
$ CHANGE 9110J-1 9110P-1
```

This changes the description of the form from its current value to the above.

```
$ CHANGE/D 9110P-1 'ONE PART 9 1/2 BY 11 UNRULED'
```

CLI *FORMSCLI Command*

Swaps to the CLI program.

Format

CLI

What It Does

This command SWAPs you to the Business BASIC CLI so that you can perform CLI commands without exiting FORMSCLI. Enter POP while in the CLI and you will be returned to FORMSCLI.

How To Use It

Execute CLI by entering it at the FORMSCLI prompt ($).

Example

You SWAP to CLI and then enter a POP to return to FORMSCLI.

```
$ CLI
!POP
$
```

CREATE

FORMSCLI Command

Creates the FORMS file.

Format

CREATE

What It Does

This command creates and initializes the FORMS file. This command should be issued once. Do this before any other FORMSCLI commands are attempted.

After creating the FORMS file, add a nonexistent form name that will never be used in a control record. This causes the spooler to ignore queue files that will be printed on devices assigned to this form name and thus allows the operator to change the form on the device.

How To Use It

Execute CREATE by entering it at the FORMSCLI prompt ($).

Example

This example creates the FORMS file and adds the form SUSPEND. SUSPEND is not used in a control record, so it halts output to a device and allows the operator to change the form mounted on the device.

```
$ CREATE
$ ADD SUSPEND 'DUMMY ENTRY TO CHANGE FORMS'
```

DELETE *FORMSCLI Command*

Deletes entries from the FORMS file.

Format

DELETE form-name

Arguments

form-name — The name of a form to be deleted from the FORMS file. Any character other than single quote (') or blank is allowed.

What It Does

Allows the user to remove entries from the FORMS file.

How To Use It

Execute DELETE by entering it at the FORMSCLI prompt ($).

Example

This deletes the form named 9110P-1 from the FORMS file.

```
$ DELETE 9110P-1
```

DISPLAY *FORMSCLI Command*

Displays the contents of the FORMS file.

Format

DISPLAY

What It Does

DISPLAY lets the user examine the contents of the FORMS file.

How To Use It

Execute DISPLAY by entering it at the FORMSCLI prompt ($).

Example

Below is an example DISPLAY output.

```
$ DISPLAY

FORM NAME     D E S C R I P T I O N                USAGE

9110PP-1      ONE PART 9 1/2 X 11 UNRULED 20#        0
9513P-2       TWO PART 9 1/2 X 11 RULED              0
DEFAULT       DEFAULT FORM N261411D                  0
N261411D-1    ONE PART 14 7/8 X 11                   0
N61011-1      ONE PART 10 5/8 X 11                   0
SUSPEND       DUMMY ENTRY TO CHANGE FORMS            0
```

FORMSCHG *FORMSCLI Command*

Swaps to the FORMSCHG program.

Format

FORMSCHG

What It Does

This causes you to swap to FORMSCHG so that you can perform FORMSCHG commands without exiting FORMSCLI. Enter POP while in FORMSCHG and you return to FORMSCLI.

How To Use It

Execute FORMSCHG by entering it at the FORMSCLI prompt ($).

Example

You swap to FORMSCHG and then enter a POP to return to FORMSCLI.

```
$ FORMSCHG
% POP
$
```

POP *FORMSCLI Command*

Terminates the FORMSCLI and clears the common area.

Format

POP

What It Does

POP terminates Business BASIC's FORMSCLI program and returns you to the previous level. POP clears the common area, which is used to pass arguments.

How To Use It

Execute POP by entering it at the FORMSCLI prompt ($).

Example

POP returns the user to keyboard mode.

```
* RUN "FORMSCLI
$ POP
*
```

QUIT *FORMSCLI Command*

Terminates FORMSCLI and retains the common area.

Format

QUIT

What It Does

QUIT terminates the FORMSCLI program retaining the common area and returns you to the previous level.

How To Use It

Execute QUIT by entering it at the FORMSCLI prompt ($).

Example

QUIT returns the user to keyboard mode.

```
* RUN "FORMSCLI
$ QUIT
*
```

SPCLI *FORMSCLI Command*

Swaps to the SPCLI program.

Format

SPCLI

What It Does

This will cause you to swap to SPCLI so that you can perform SPCLI commands without exiting FORMSCLI. Enter POP while in SPCLI and you will be returned to FORMSCLI.

How To Use It

Execute SPCLI by entering it at the FORMSCLI prompt ($).

Example

You SWAP to SPCLI and then enter a POP to return to FORMSCLI.

```
$ SPCLI
# POP
$
```

QTYPE

Spooler Utility

Displays a queue file.

Format

```
{RUN  }
{SWAP } "QTYPE
{CHAIN}
```

What It Does

Use the Business BASIC utility QTYPE to display a queue file on your terminal. To use this command, you must be in the $SPL ($SPL3) directory or have a link to QTYPE from another directory.

How To Use It

To execute QTYPE, enter RUN, CHAIN, or SWAP "QTYPE.

Example

Enter queue filename TRI$5538 when QTYPE prompts you for a filename.

```
* RUN "QTYPE
FILENAME: TR1$5538
```

Spooler CLI Commands (SPCLI)

Spooler Utility

Maintains table of queue names and device assignments.

Format

```
{ RUN   }
{ SWAP  }"SPCLI
{ CHAIN }
```

What It Does

The SPCLI is the control program that maintains the table of queue names and device assignments. It also provides a means for printing queue files on a demand basis. SPCLI commands are used like BASIC CLI commands. In addition, auto-indirect (macro) files are supported. If a command is not recognized by the SPCLI, a search of the $SPL ($SPL3) directory is made for a file named command.SP and, if such a file is found, it is processed as a command file.

The SPCLI commands are:

Command	Action
BMAX	Sets or displays the maximum number of BATCH jobs.
BYE	Logs you off the system.
CLI	Swaps to the CLI program.
DELETE	Deletes a queue file from the spool system.
DEV	Lists the current device assignments.
FORMS	Changes the current FORMS assignment.
FORMSCHG	Swaps to the FORMSCHG program.
FORMSCLI	Swaps to the FORMSCLI program.
LINK	Assigns queue names to output devices.
LOG	Lists the contents of the error summary file.
PMAX	Sets or examines the maximum number of PRINTER jobs.

Spooler CLI Commands (SPCLI) *Continued*

Command	Action
POP	Terminates the SPCLI and clears the common area.
PRINT	Forces printing of a queue file on a demand basis.
PRIORITY	Sets the output device's priorities.
PROC	Changes the default QTABLE.
QUES	Lists the queue files presently in the spool system.
QUIT	Terminates the SPCLI and retains the common area.
RELINK	Changes the device assignments for a queue name.
RESTORE	Restores queue files for reprinting.
START	Logs a spooler job onto the system.
STAT	Lists spooler jobs running.
SUSPEND	Interrupts a spooler.
UNLINK	Deletes device assignments for queue names.

How To Use It

To execute SPCLI, enter RUN, CHAIN, or SWAP "SPCLI.

When you run the SPCLI, the prompt is the pound sign (#).

BMAX *SPCLI Command*

Sets or displays the maximum number of BATCH jobs.

Format

BMAX [maximum-jobs]

Arguments

maximum-jobs The maximum number of batch jobs a spooler can start or 0.

What It Does

If no argument is specified, BMAX displays the maximum number of BATCH jobs. If maximum-jobs is set to zero, the spooler swaps to BATCH to execute the batch job stream. If maximum-jobs is greater than zero, the spooler starts another job for each batch request as long as the current number of BATCH jobs is less than maximum-jobs.

The number of BATCH jobs is limited by the number of available jobs. If a spooler tries to start a BATCH job when no jobs are available, it waits until a job logs off. Also, each BATCH job starts another job that executes the commands in the batch queue file; thus, each batch request requires one job if BMAX equals 0 or two jobs if BMAX is nonzero.

How To Use It

Execute BMAX by entering it at the SPCLI prompt (#).

Example

```
# BMAX 1
```

The maximum number of BATCH jobs that can be started is set to 1.

BYE *SPCLI Command*

Logs you off the system.

Format

BYE

What It Does

BYE terminates the SPCLI program and logs you off Business BASIC.

How To Use It

Execute BYE by entering it at the SPCLI prompt (#).

Example

```
# BYE
```

CLI

SPCLI Command

Swaps to the CLI program.

Format

CLI

What It Does

This command swaps you to the CLI so that you can execute CLI commands without exiting the SPCLI. Entering POP or QUIT while in CLI returns you to the SPCLI.

How To Use It

Execute CLI by entering it at the SPCLI prompt (#).

Example

SWAP to CLI, initialize a tape drive, and then return to SPCLI.

```
# CLI
!INIT MT0;POP
#
```

DELETE *SPCLI Command*

Deletes a queue file from the spooler system.

Format

DELETE[global-switches] queue-file1 ...

Arguments

queue-file — The unique user spool file in $SPL ($SPL3) that is to be deleted.

Global Switches

/A — Delete the queue file even if it is protected.

/V — Verify deletions.

What It Does

DELETE deletes the named queue files from the SPOOL system.

How To Use It

Execute DELETE by entering it at the SPCLI prompt (#).

Example

This deletes queue file LPT$6043, if it exists.

```
# DELETE LPT$6043
```

DEV *SPCLI Command*

Lists the current device assignments.

Format

DEV[global-switches]

Global Switches

/L — List the current device assignments to the default output queue. Use the Business BASIC CLI GQUE command to retrieve the default output queue.

What It Does

DEV displays the current state of the queue table, including queue names, device names, form names, priorities, and status switches. The status field is established by the SPCLI command LINK. Note that the status field is filled with asterisks (*) if more than five switches are set for the queue name; however, the status in the queue table is unaffected.

How To Use It

Execute DEV by entering it at the SPCLI prompt (#).

Example

Four assignments are made: LPT to the line printer loaded with the form APCHECK and printing all queue file priorities; LP2 to a printer with form feed on port 2 with no forms loaded and all priorities; TR to a printer without form feed on port 12 with form N9513-2, the status S returned from the device, priorities from 0 (high) to 127 (low), and queue files to be retained for one day after printing; and LPX to a terminal printer with form feed on port 4 with the default form DEFAULT loaded and all priorities.

```
# DEV

 QTABLE0              01/25/85          13:41:47
RECORD   QNAME   DEV    STATUS      PRIORITY          FORMS
                                    HI      LO

   1       LPT   $LPT                 0     255       APCHECK
   2       LP2   $PR2                 0     255       SUSPEND
   3       TR    $TR12      S         0     127       N9513-2
   4       LPX   $PR4                 0     255       DEFAULT
```


FORMS *SPCLI Command*

Changes the current FORMS assignment.

Format

FORMS device-name form-name

Arguments

device-name — Any output device supported by Business BASIC.

form-name — The name of a form existing in the FORMS file that you want to assign to device-name.

What It Does

This command changes the form-name currently assigned to the output device.

How To Use It

Execute FORMS by entering it at the SPCLI prompt (#).

While changing forms, assign the device a nonexistent form that has been defined in the FORMS file for that purpose. This prevents the spooler from starting the device while the new forms are being mounted.

Example

This changes the form assignment for the line printer, $LPT, to the form-name INVOICE, if the form has been set up in the forms control file FORMS. This command will change the forms entry for all queue names linked to the device.

```
# FORMS $LPT INVOICE
```

FORMSCHG *SPCLI Command*

Swaps to the FORMSCHG program.

Format

FORMSCHG

What It Does

This command swaps to FORMSCHG so that you can perform FORMSCHG commands without exiting SPCLI. Enter POP while in FORMSCHG to return to SPCLI.

How To Use It

Execute FORMSCHG by entering it at the SPCLI prompt (#).

Example

Swap to FORMSCHG and then enter a POP to return to SPCLI.

```
# FORMSCHG
% POP
#
```

FORMSCLI *SPCLI Command*

Swaps to the FORMSCLI program.

Format

FORMSCLI

What It Does

This command swaps you to FORMSCLI so that you can perform FORMSCLI commands without exiting SPCLI. Enter POP while in FORMSCLI to return to SPCLI.

How To Use It

Execute FORMSCLI by entering it at the SPCLI prompt (#).

Example

This swaps you to FORMSCLI, and then you enter a POP to return to SPCLI.

```
# FORMSCLI
$ POP
#
```

LINK *SPCLI Command*

Assigns queue names to output devices.

Format

LINK[global-switches] [queue-name] [device-name][local-switches] ...

Arguments

queue-name — One to five alphanumeric characters used to describe a device name.

device-name — An output device supported by Business BASIC.

Global Switches

/A — Duplicate the queue name if it is already in the queue table.

/V — Verify the assignments as they are made.

Local Switches

These switches are associated with the queue-name/device pair. Place them on the device argument of the command.

/B — Interpret queue files as command files for the batch processor rather than as print queue files.

/D — Delete the queue file without printing it.

/E — Execute a form feed both before and after printing.

/F — Execute a form feed on a line printer after printing rather than before printing.

/N — Suppress form feed at end of print.

/R — Retain the queue file sent to this queue-name after printing.

/S — Retain the queue file sent to this queue-name for one day after printing.

The local /N switch suppresses the form feed at the end of print processing. If the local /N switch is used with the /F switch, form feeds are suppressed at both the beginning and end.

LINK *Continued*

What It Does

LINK places an entry in the queue table assigning a device to queue-name so that queue files matching queue-name are printed on device if it is available. If device is not idle or not in the system, no error is issued, but no queue files are output. When you use the DEV command, the STATUS column displays the local switches that were assigned to the device when LINK was executed. (See the DEV command example).

How To Use It

Execute LINK by entering it at the SPCLI prompt (#).

Example

This example assigns the queue name APCHK to the line printer, $LPT, and the queue name LP7 to a terminal printer with form feed on Business BASIC port 7 and the local switches /N (suppress form feed at end of print) and /S (queue files sent to this queue name will be retained for one day after printing). The global /V switch is used so that the assignments are verified after they are made. The SPCLI command DEV was entered after the LINK so that the assignments can be checked. Note that the device $PR7 has no forms assigned (indicated by the ...). To assign the device $PR7 a form, you must use the SPCLI command FORMS described earlier.

```
# LINK/V APCHK $LPT LP7 $PR7/N/S
LINKED: APCHK:$LPT
LINKED: LP7:$PR7
# DEV

 QTABLE0              01/24/85          13:41:47
RECORD   QNAME    DEV    STATUS      PRIORITY          FORMS
                                     HI      LO

   1       LPT    $LPT                 0     255        DEFAULT
   2       LP2    $PR2                 0     255        SUSPEND
   3       TR     $TR12      S         0     127        NP513-2
   4       APCHK  $LPT                 0     255        DEFAULT
   5       LP7    $PR7      NS        0     255     ..........
```

LOG *SPCLI Command*

Lists the contents of the error summary file.

Format

LOG[global-switches]

Global Switches

/D	Delete the error file (no listing produced).
/L	List the contents of the error summary file to the default output queue. Use the Business BASIC CLI command GQUE to retrieve the default output queue.

What It Does

LOG lists the current contents of the error summary file, PRINTLOG, including error messages and SPOOL system messages placed there.

How To Use It

Execute LOG by entering it at the SPCLI prompt (#).

Example

In this example two messages are listed: the daily file delete message from the SPCLEANUP program and an error referring to queue file LP$1234. The time and date the message was logged are included along with the job number of the spooler in case multiple spoolers are running.

```
# LOG

SPOOL SYSTEM FILE DELETE - COMPLETED  (NO FILES DELETED)
          9:54:55                      01/25/85
ERROR IN OPENING QUE FILE: LP$1234
          SPOOL JOB # 02          13:44:50          01/25/85
```


PMAX *SPCLI Command*

Sets or examines the maximum number of PRINTER jobs.

Format

```
PMAX [maximum-jobs]
```

Arguments

maximum-jobs The maximum number of printer jobs a spooler can start or 0.

What It Does

If no argument is specified, PMAX displays the current maximum number of PRINTER jobs. If maximum-jobs is set to zero, the spooler will swap to PRINTER to output a queue file. If maximum-jobs is greater than zero, the spooler will start another job for each queue file to be output as long as the current number of PRINTER jobs is less than maximum-jobs.

The number of PRINTER jobs is limited by the number of jobs generated into Business BASIC. If a spooler tries to start a PRINTER job when no jobs are available, it will wait until a job logs off. Since no two PRINTER jobs can output to a device simultaneously, the value of PMAX need never be greater than the number of output devices defined in the queue table.

How To Use It

Execute PMAX by entering it at the SPCLI prompt (#).

Example

This sets the maximum number of PRINTER jobs to 3.

```
#PMAX 3
```

POP

SPCLI Command

Terminates the SPCLI and clears the common area.

Format

```
POP
```

What It Does

POP terminates the SPCLI program and returns you to the previous level. POP clears the common area, which is used to pass arguments.

How To Use It

Execute POP by entering it at the SPCLI prompt (#).

EXAMPLE

```
# POP
```

PRINT *SPCLI Command*

Forces printing of a queue file on a demand basis.

Format

PRINT[global-switches] [job-number/J] [copies/C] [device-name/D] queue-file1 ...

Arguments

job-number/J	The job number of the spooler to be activated.
copies/C	The number of copies of each queue file to be printed.
device-name/D	The name of the output device to be used.
queue-file	The unique user spool file in $SPL ($SPL3) that is to be printed.

Global Switches

/A	Force retained queue files to be printed.
/M	SPOOLER returns a response upon completion.
/S	Retain the queue file after printing.
/W	SPOOLER awaits another command, up to 2 minutes.

What It Does

PRINT suspends (interrupts) either the lowest-numbered SPOOLER job or the job identified by the local /J switch if this job is a SPOOLER. PRINT forces the specified queue file to be printed either on the device linked to the queue name that was specified in the user program, or the device identified by the local /D switch, if this device is available. Local switches other than /J apply until they are encountered in the argument list again. (See also the note for the SPCLI command SUSPEND).

How To Use It

Execute PRINT by entering it at the SPCLI prompt (#).

Example

This interrupts the spooler with the lowest job number and causes three copies of queue file CRJ$4088 to be printed on the line printer.

```
# PRINT 3/C $LPT/D CRJ$4088
```


PRIORITY *SPCLI Command*

Sets the output device's priorities.

Format

PRIORITY device-name low/L high/H

Arguments

device-name	Any output device supported by Business BASIC.
low/L	A numeric argument representing the low priority for this device. The lowest priority is 255.
high/H	A numeric argument representing the high priority for this device. The highest priority is 0.

What It Does

PRIORITY sets the priority range for the output device. Only queue files with a priority falling within the range for the output device are printed.

How To Use It

Execute PRIORITY by entering it at the SPCLI prompt (#).

Example

Set the priority range for the line printer, $LPT, to 0 (high) through 63 (low).

```
# PRIORITY $LPT 63/L 0/H
```

PROC *SPCLI Command*

Selects the queue table to be referred to by SPCLI.

Format

PROC processor-number

Arguments

processor-number	A 0 refers to a QTABLE0 in the background and a 1 refers to a QTABLE1 in the foreground.

What It Does

When SPCLI is executed, the queue table name referred to is determined by SYS(25), QTABLE0 for the background and QTABLE1 for the foreground. PROC changes the queue table name currently referred to by SPCLI to QTABLE<processor-number>. The selected queue table name remains in effect for the duration of this SPCLI session or until another PROC command is issued. Note, PROC has no effect on other spooler utilities. PROC allows both queue tables, QTABLE0 and QTABLE1, to be referred to from one ground for redefining queue-name/device-name links and other queue table information. If the selected queue table does not exist, it will be created.

How To Use It

Execute PROC by entering it at the SPCLI prompt (#).

Use the DEV command to display the QTABLE number.

PROC *Continued*

Example

The first DEV command shows the default QTABLE0, because you are running in the background. The SPCLI PROC command is then used to change the queue table to QTABLE1. The next DEV command shows the assignments made in QTABLE1.

```
# DEV

  QTABLE0                01/24/85          13:41:47
 RECORD    QNAME     DEV    STATUS        PRIORITY          FORMS
                                          HI      LO

    1        LPT     $LPT                  0     255        DEFAULT
    2        APCHK   $LPT                  0     255        DEFAULT

# PROC 1

# DEV

  QTABLE1                01/24/85          13:44:27
 RECORD    QNAME     DEV    STATUS        PRIORITY          FORMS
                                          HI      LO

    1        LP2     $PR2                  0     255        SUSPEND
    2        TR      $TR2        S         0     255        DEFAULT
```

QUES *SPCLI Command*

Lists the queue files presently in the spool system.

Format

QUES[global-switches] [queue-file1[local-switches]]...

Arguments

queue-file	The unique user spool file in $SPL ($SPL3) that is to be listed.

Global Switches

/A	Include the queue files that were retained after printing. P will be displayed in the status column.
/E	Include extended queue file information.
/L	List to the default output queue. Use the Business BASIC CLI GQUE command to retrieve the default output queue.
/Q	Quick listing, four queue files across the page.
/S	Produce a sorted listing.

Local Switches

/N	Do not include files matching the argument.
/X	Require an exact match on the argument.

What It Does

QUES lists the names of the unprotected queue files that are presently in the SPOOL system. Queue file names are prefixed with an asterisk if currently open. For search purposes, queue file names are normally suffixed with "$-" to obtain a match on all queue files with the specified name. The local /X switch overrides this. Header information from the queue file can also be listed.

How To Use It

Execute QUES by entering it at the SPCLI prompt (#).

QUES *Continued*

Example

```
# QUES/A/E
```

Produce an extended listing of all queue files, including protected files.

```
  QFILES       01/25/85          11:34:46

P TEST$5739    01/25/85  15:56   AAJLE6  LIST          958 BYTES
  TR$4148      01/25/85  11:31   ABWEB6  CLI           0 BYTES
* PR$4167      01/25/85  11:34                         -512 BYTES
* TR$4154      01/25/85  11:32   ABWEB6  CLI           16551 BYTES
  PR$4158      01/25/85  11:33   ABWEB6  TESTGEN       295 BYTES

a    b            c       d        e       f             g
```

The fields are as follows:

a) Status of the queue file:

- P Queue file retained after printing (Business BASIC CLI command CHATR designates write protection with a /W).
- * Use count greater than 0 (file is currently open).

b) Name of the queue file.

c) Creation date.

d) Creation time.

e) Account code of the user who created the queue file.

f) Program that created (opened) the queue file.

g) File size (-512 for a file that has just been created).

QUIT *SPCLI Command*

Terminates the SPCLI and retains the common area.

Format

QUIT

What It Does

QUIT terminates the SPCLI program retaining the common area and returns you to the previous level.

How To Use It

Execute QUIT by entering it at the SPCLI prompt (#).

Example

This example shows the SPCLI entered from keyboard mode. Keyboard mode is returned to by entering QUIT.

```
* RUN "SPCLI
$ QUIT
*
```

RELINK *SPCLI Command*

Change the device assignments for a queue name.

Format

RELINK[global-switches] queue-name1 device-name1[local-switches] ...

Arguments

device-name	An output device supported by Business BASIC.
queue-name	One to five alphanumeric characters used to describe a device-name.

Global Switches

/F	Change only the first occurrence of the queue name; any remaining occurrences are left intact.
/V	Verify the changes.

Local Switches

These switches are associated with the queue-name/device pair. Place them on the device argument of the command.

/B	Batch queue; queue files are interpreted as command files for the batch processor rather than as print queue files. The batch facility is described elsewhere in this chapter.
/D	Queue file is not printed and is deleted.
/E	Execute a form feed both before and after printing.
/F	Form feed on line printer occurs following printing rather than before printing.
/N	No form feed at the end of printing.
/R	Retain the queue file sent to this queue-name after printing.
/S	Retain the file sent to this queue-name for one day after printing.
/*	Retain current status information.

The local /N switch suppresses the form feed at the end of print processing. If /* is used with the /F switch, form feeds are suppressed at both the beginning and end.

RELINK *Continued*

What It Does

RELINK changes the queue table device assignment for the specified queue name to the specified device. By default, the first occurrence of the queue name is changed, and all remaining occurrences are deleted. This can be altered with the global /F switch to retain the remaining occurrences. The local /* switch retains the current status information and includes any new information as well.

How To Use It

Execute RELINK by entering it at the SPCLI prompt (#).

Example

The first DEV command diplays the assignments before the SPCLI command RELINK is used. RELINK changes the assignment of the queue name TR to a printer without form feed on Business BASIC port 5, retaining any status switches originally selected, as can be seen with the second DEV command.

```
# DEV

  QTABLE0              01/24/85          13:41:47
 RECORD    QNAME    DEV    STATUS       PRIORITY          FORMS
                                        HI      LO

    1        LPT    $LPT                 0     255        DEFAULT
    2        LP2    $PR2                 0     255        SUSPEND
    3        TR     $TR12       S        0     127        NP513-2
    4        APCHK  $LPT                 0     255        DEFAULT
    5        LP7    $PR7       NS        0     255        DEFAULT

# RELINK TR $TR5/*
# DEV

  QTABLE0              01/24/85          13:41:47
 RECORD    QNAME    DEV    STATUS       PRIORITY          FORMS
                                        HI      LO

    1        LPT    $LPT                 0     255        DEFAULT
    2        LP2    $PR2                 0     255        SUSPEND
    3        TR     $TR5        S        0     127        NP513-2
    4        APCHK  $LPT                 0     255        DEFAULT
    5        LP7    $PR7       NS        0     255        DEFAULT
```

RESTORE *SPCLI Command*

Restores queue files for reprinting.

Format

RESTORE[global-switches] queue-file1 ...

Arguments

queue-file — The unique user spool file in $SPL ($SPL3) that is to be restored.

Global Switches

/V — Verify the restorations.

What It Does

RESTORE restores the specified queue files to unretained condition for reprinting. After the queue file has been printed, the status of the queue file returns to retained.

How To Use It

Execute RESTORE by entering it at the SPCLI prompt (#).

Example

```
# RESTORE LPT$5147
```

This changes the status of queue file LPT$5147 to unretained. The file is printed if LPT is linked to an available device. After the file has been printed, the queue file LPT$5147 is retained again.

START *SPCLI Command*

Logs a spooler job on to the system.

Format

START

What It Does

START assigns the first available job, if any, to a spooler and logs it on to the system.

How To Use It

Execute START by entering it at the SPCLI prompt (#).

Example

```
# START
```

STAT *SPCLI Command*

Lists spooler jobs running.

Format

STAT[global-switches]

Global Switches

/A Include all jobs in the listing.

/R Repeat the listing until interrupted (IKEYed).

What It Does

STAT lists the job number, account, directory, program name, and port number of all spooler jobs. If no switches are supplied and only the header is printed, then there are no spoolers running.

This command is not the same as the Business BASIC STAT utility.

How To Use It

Execute STAT by entering it at the SPCLI prompt (#).

Example

This example indicates one spooler running on job 2, detached in directory $SPL under account SPOOL. The * in the account indicates a started job.

```
# STAT

  JOB      ACCNT          DIR            PROGRAM          PORT

   2      SPOOL*         $SPL           SPOOLER           -1
```


SUSPEND *SPCLI Command*

Interrupts a spooler.

Format

SUSPEND[global-switches] [job-number]

Arguments

job-number	Actual job number of the spooler. (See the SPCLI command STAT.)

Global Switches

/D	Delete the queue file if one is printing.
/K	Log the spooler off the system after stopping output.
/M	Spooler returns a response when complete.
/R	Allow the queue file to reprint normally if one was printing.
/W	Force the spooler to wait for another command, up to 2 minutes.

What It Does

SUSPEND interrupts the lowest-numbered inactive spooler job or the specified job-number if it is a SPOOLER. If an active spooler job is to be interrupted, the job-number argument must be used. If a queue file is printing when a spooler is suspended, the output stops and the queue file is protected and saved by default.

If further spooler action is to follow, the global /W switch can be used.

How To Use It

Execute SUSPEND by entering it at the SPCLI prompt (#).

Find the job number by using the SPCLI command STAT.

Example

This interrupts and logs off the inactive spooler with the lowest job number. A message is logged in PRINTLOG.

```
# SUSPEND/K
```

UNLINK *SPCLI Command*

Deletes device assignments for queue names.

Format

UNLINK [global-switches] { queue-name / device/D } [device/D] queue-name . . .

Arguments

queue-name	A name representing a device queue.
device/D	Argument is a device name rather than a queue name.

Global Switches

/A	Clear the queue table file.
/F	Delete only the first occurrence of the queue name.
/V	Verify deletions as they are made (except in the case of /A).

What It Does

UNLINK causes all queue names associated with a device to be unlinked.

UNLINK removes from the queue table file all device assignments for the specified queue names. The global /A switch provides a means of clearing the entire queue table, i.e., removing all queue name assignments (see also RELINK).

How To Use It

Execute UNLINK by entering it at the SPCLI prompt (#).

Example

The first UNLINK removes the device assignment for queue name TR from the queue table. The second UNLINK removes all assignments from the queue table for the device $PR2.

UNLINK *Continued*

```
# DEV

  QTABLE0              01/28/85          14:41:47
 RECORD    QNAME    DEV    STATUS       PRIORITY          FORMS
                                        HI      LO

    1        LPT    $LPT                 0      255        DEFAULT
    2        LP2    $PR2                 0      255        DEFAULT
    3        TR     $TR12      S         0      127        NP513-2
    4        APCHK  $PR2                 0      255        DEFAULT

# UNLINK TR
# DEV

  QTABLE0              01/28/85          14:42:47
 RECORD    QNAME    DEV    STATUS       PRIORITY          FORMS
                                        HI      LO

    1        LPT    $LPT                 0      255        DEFAULT
    2        LP2    $PR2                 0      255        DEFAULT
    3        APCHK  $PR2                 0      255        DEFAULT

# UNLINK/V $PR2/D
UNLINKED: $PR2
# DEV

  QTABLE0              01/28/85          14:44:27
 RECORD    QNAME    DEV    STATUS       PRIORITY          FORMS
                                        HI      LO

    1        LPT    $LPT                 0      255        SUSPEND
```

VACUUM

RDOS Business BASIC Utility

Creates RDOS CLI macros to work with subdirectories and partitions.

Format

```
{RUN  }
{SWAP }"VACUUM
{CHAIN}
```

What It Does

The Business BASIC utility VACUUM creates the RDOS CLI macro files to perform operations on the subdirectories and subpartitions on a disk.

VACUUM is useful for creating CLEAR macros to use after an RDOS system crash. The CLEAR macros locate and clear the use counts of files that were open at the time of the crash.

VACUUM macros can specify RDOS CLI operations other than CLEAR, such as LIST, MOVE, or DUMP. When VACUUM creates the macro, use an editor to make any changes.

How To Use It

To execute VACUUM, enter RUN, CHAIN, or SWAP "VACUUM.

VACUUM prompts you for the directory or partition where the CLEAR is to begin. It then prompts you to select whether you want to sort the macro file to list the directories in alphabetical order. VACUUM then prompts you to select whether you want to use CLEAR, LIST, or another command on the indirect file. Then you're prompted for the output file name, which is the name of the macro. Other prompts depend on whether you requested to use CLEAR, LIST, or another command on the indirect file.

- If you selected CLEAR, you are prompted to select verification and whether or not the device should be included.
- If you selected LIST, you are prompted to select whether you want a sorted list and whether you want output sent to the printer.
- If you selected OTHER, you are prompted to enter the commands. Enter one or more RDOS CLI commands with the corresponding switchs. Separate each command with a semicolon.

VACUUM *Continued*

Example

```
* RUN "VACUUM"
VACUUM Ver X.XX

STARTING PARTITION: DZ0
```

Enter the directory or partition from which to start the CLEAR operation. The system then displays a list of all directories under the directory or partition you specified.

```
SORT MACRO FILE? YES
```

Enter YES to list and alphabetize the directories in the macro.

```
SORTING - PLEASE WAIT

INDIRECT FILE TO CLEAR, LIST OR OTHER? CLEAR
```

Enter OTHER and then the command name for operations other than CLEAR or LIST.

```
OUTPUT FILE NAME? CLEARDRS.MC
```

Use .MC for an automatic macro. Request below that the cleared files be verified and that devices be cleared also.

```
VERIFY? Y
DEVICES? Y


* !TYPE CLEARDRS.MC
```

The CLEARDRS.MC macro that VACUUM created is shown below.

```
DIR DZ0:$DOC;GDIR;CLEAR/A/V;CLEAR/A/D;RELEASE $DOC
DIR DZ0:$LIB;GDIR;CLEAR/A/V;CLEAR/A/D;RELEASE $LIB
DIR DZ0:$SPL;GDIR;CLEAR/A/V; CLEAR/A/D;RELEASE $SPL
DIR DZ0:$SYS;GDIR;CLEAR/A/V;CLEAR/A/D;RELEASE $SYS
DIR DZ0:BASICGEN;GDIR;CLEAR/A/V;CLEAR/A/D;RELEASE BASICGEN
DIR DZ0;GDIR;CLEAR/A/V;CLEAR/A/D
```

To execute this macro, enter CLEARDRS at the RDOS CLI R prompt.

End of Chapter

Chapter 7
Privileged System Calls

The Business BASIC system manager (an AA account) uses privileged system calls to access portions of memory and alter memory contents, interrupt and log off jobs, tune system resources, execute operating system calls and special programs, and send messages to all terminals.

Before using any privileged system calls to execute operating system calls, you should be familiar with the operating system call you want to use. For more information, consult the operating system manual in which system calls are described. Some packets are different depending on whether you are issuing a 32-bit system call or a 16-bit system call.

Only AA accounts can use these privileged system calls, but any user can execute a program that contains these system calls. You can allow all users to enter these commands. In AOS, make the choice during system generation. In RDOS, use a switch at system execution.

Many of these system calls apply only to Business BASIC under a particular operating system. To aid users converting from one operating system to another, privileged system call numbers that are not used in one operating system (for example, "Not used in AOS") are distinguished from privileged system call numbers that are not used in either system (that is, "Not used").

AOS Privileged System Calls

AOS privileged system calls are STMBs, STMCs, and STMEs. The AOS system still provides security with the access control lists and the PREDITOR utility. Remember, any user can execute programs that contain privileged system calls.

STMBs in AOS Business BASIC

The STMB calls allow you to access and modify portions of memory. You must be extremely careful when you alter memory contents to be sure that you are actually modifying the locations you want. The general format of STMB statements and commands is:

```
STMB call, arguments
```

where call is the number of the STMB call and arguments vary with each call. Individual STMBs are explained on the following pages.

STMB 0, item, address

Retrieves a word address, where address is a variable receiving the word address of item. Items that are undefined return a value of 65535.

Item	Description
0	Returns the memory address of a 21-element array. Each element of this array is the address returned by the corresponding STMB 0,item,address call.
1	Not used in AOS.
2	Not used in AOS.
3	Not used in AOS.
4	Not used.
5	Not used.
6	Returns the memory address of a word containing the address of the user channel table. The channel table contains four words for each of the 16 user channels that can be opened. The first four words apply to user channel 0; the next four words apply to user channel 1; etc. For each channel number, the meanings of the words are: • system channel number • mode of OPEN • current file position - high • current file position - low This information corresponds to the first two columns of information returned by the utility program SCHANS.
7	Not used in AOS.
8	Returns the memory address of a word containing the maximum memory for the current job.
9	Returns the memory address of a two-element array that represents the global switches on Business BASIC.
10	Not used in AOS.
11	Not used.

Item	Description
12	Not used in AOS.
13	Not used.
14	Not used in AOS.
15	Returns the memory address of a double-word that is used by SYS(29).
16	Not used.
17	Not used.
18	Not used.
19	Returns the memory address of a word containing the address of the current user's User Status Table.
20	Not used.
21	Returns the address of the physical channel number for the push file (BASIC.PS).

STMB 1, wordcount, address, destination

Copies the contents of memory starting at address (word address) for the number of words specified in wordcount to the variable or array destination.

STMB 2, wordcount, address, source

Copies the number of words specified in wordcount from the variable or array source into memory at the address (word address) specified.

STMB 3, address, destination$

Copies bytes from memory beginning at address into the string or substring variable destination$ until destination$ is full. In AOS, a byte address (word address * 2) is supplied for the address argument. In AOS/VS, a word address is supplied. This word address is converted to a byte address by Business BASIC.

STMB 4, address, source$

Copies the entire contents of a string or substring variable source$ into memory starting at the address specified. In AOS, a byte address (word address * 2) is supplied for the address argument. In AOS/VS, a word address is supplied. This word address is converted to a byte address by Business BASIC.

STMB 5, wordsize, address, destination (AOS/VS only)

Copies the contents of memory into the destination variable, starting at address (word address) for the number of words specified in wordsize (1 or 2). The wordsize argument must be 1 (for a narrow word) or 2 (for a double word). If any other value is used, error 37 ("User Routine") occurs.

Use STMB 5 when you want to store a double word value in the destination variable.

STMB 6, wordsize, address, source (AOS/VS only)

Copies from the variable source into memory the number of words specified in wordsize (1 or 2), starting at the word address specified. The wordsize argument must be 1 (for a narrow word) or 2 (for a double word). If any other value is used, error 37 ("User Routine") is triggered.

Use STMB 6 when you want to retrieve a double word value from a source variable.

STMB 7

Not used in AOS.

STMB 8

Not used in AOS.

STMB 9

Not used.

STMB 10, function, address, mask

If function = 1, STMB 10 sets the bits in the word at the specified word address according to the bits set in the variable mask. If function = 0, STMB 10 clears the bits in the word at the specified address according to the bits set in the variable mask.

STMB 11, string$, variable1, variable2, variable3

Scans the contents of string$, puts the number of 1 bits in variable1, and puts the largest consecutive number of 0 bits in variable2. Variable3 is required and is set to 0. String$ may be a substring, but it must be word-aligned and of even byte length—that is, the subscripts of string$[x,y] must be such that x is odd and y is even.

STMB 12

Not used in AOS.

STMB 13

Not used in AOS.

STMB 14

Not used in AOS.

STMB 15

Not used in AOS.

STMB 16, flag

Sets (flag = 1) or clears (flag = 0) the run-only flag in the current program. If set, this flag prevents a file from being listed or modified by non-AA accounts.

STMB 17

Performs an STMB 20.

STMB 18

Not used in AOS.

STMB 19

Not used in AOS.

STMB 20

Performs an immediate BYE to log off the current job, completes system housekeeping, and returns to the previous level.

STMB 21

Not used.

STMB 22, byte-address, string-variable

Returns the byte address of the string variable.

STMB 23, address, variable

Returns the word address of the numeric variable in address.

STMB 24, error, accumulator [,flag]

Performs a system call using the information in the accumulator string. The optional flag argument consists of bit flags modifying the other arguments. The value returned in error is a system error code or zero.

NOTE: Do not use STMB 24 to make system calls that can be made with an existing Business BASIC statement, utility, or command. You should particularly avoid the use of STMB 24 system calls to manipulate files that are already being manipulated through the use of Business BASIC statements. Mixing the use of STMB 24 with Business BASIC statements can cause unpredictable, and probably undesirable, results.

For example, if you use the OPEN FILE statement to open a file, you should not close the file by using STMB 24 to issue a ?CLOSE call. You should instead use the CLOSE FILE statement. Similarly, you should use the POSITION FILE statement, not STMB 24, to issue a ?SPOS call to position a file pointer. In addition, when terminal type 8 is being used, do not use STMB 24 to make a ?SDLM call to change the delimiter table. Terminal type 8 sets the delimiter table for its own purposes.

In AOS Business BASIC

The accumulator (AC) string contents are listed as follows:

Bytes	Contents
1-2	Contents of AC0 for system call.
3-4	Contents of AC1 for system call.
5-6	Contents of AC2 for system call.
7-8	System call number (system dependent).
9-10	Channel number for I/O calls (this may duplicate the contents of an accumulator).

You can use only Business BASIC channel numbers, not system channel numbers. If you need an address for an accumulator, you can use STMB 22 or STMB 23 to obtain it. When I/O is being performed, the flag argument

specifies the Business BASIC channel. Depending on the flag, Business BASIC maps this number into the appropriate system channel and places the value into the proper accumulator for the call. The bit values for flag are listed as follows:

Bit	Mask	Meaning
15	1	I/O call (convert channel and put in AC2).
14	2	OPEN call returning channel number that must be equated to a Business BASIC channel.
13	4	Reserved; must be 0.
12	8	Convert channel into AC0.
11	16	Convert channel into AC1.

In AOS/VS Business BASIC

The accumulator (AC) string contents are listed as follows:

Bytes	Contents
1-4	Contents of AC0 for system call.
5-8	Contents of AC1 for system call.
9-12	Contents of AC2 for system call.
13-14	System call number.
15-16	Channel number for I/O calls.

You can use only Business BASIC channel numbers, not system channel numbers. If you need an address for an accumulator, you can use STMB 22 or STMB 23 to obtain it. When I/O is being performed, the flag argument specifies the Business BASIC channel. Depending on the flag, Business BASIC maps this number into the appropriate system channel and places the value into the proper accumulator for the call. The bit values for flag are listed as follows:

Bit	Mask	Meaning
15	1	I/O call (convert channel and put in AC2).
14	2	OPEN call returning channel number that must be equated to a Business BASIC channel.

13	4	Reserved; must be 0.
12	8	Convert channel into AC0.
11	16	Convert channel into AC1.

When using STMB 24 in conjunction with STMBs 22 and 23, remember that addresses are 4 bytes long in a 32-bit environment. To alter an address, use the SHFT function. Suppose, for example, you have used STMB 22 to retrieve the byte address of a string, but the system call requires a word address. You should use the following command to shift the address to the right: SHFT(byte-address,-1). To change a word address to a byte address, use this command: SHFT(word-address,1).

STMB 25, userchannel, systemchannel

Maps the user's channel to the system's channel, making the system channel accessible to the user. This is used to access the push file, BASIC.PS, in utilities like RNAM, VAR, and PD. Issuing a CLOSE on a user channel opened with STMB 25 does not close the physical channel but frees the user channel.

STMCs in AOS Business BASIC

STMCs allow you to use system calls. With STMCs, Business BASIC checks only for compatibility of variable types; it does not check variable length. Therefore, if a 36-byte string is required and you supply only a 20-byte string, the 16 bytes following the string will overwrite part of your program and cause undesirable results.

When you use a string literal as an argument, the literal must include a terminating null. Without the null, the operating system reads beyond the end of the literal and produces erroneous results. In order to pass the literal "ABC", you must write the literal as `ABC<0>`.

When you use a string variable as an argument to receive a value (for example, ?GNAME), fill the variable with nulls before using the STMC.

Arguments that refer to channel numbers refer to a file opened in the current Business BASIC program on this relative channel. For example, 2 refers to the file opened in the current program on channel 2.

When a user accesses a file through the standard Business BASIC statements (OPEN, READ, PRINT, CLOSE, etc.), AOS Business BASIC uses the operating system's search list mechanism.

The general format for STMC is:

STMC call, error, arguments

where call is one of the following calls (by number); error is a variable to receive the operating system error returned (a positive error code is returned, so negate it to find

it in the BASIC.ER file; -1 is returned if the call is successful); and arguments vary with each call. Individual STMCs are explained on the following pages.

STMC 0, error, filename, sectors

?CREATE creates a contiguous file with all locations initialized to 0. The maximum value of sectors is 32767.

STMC 1, error, directoryname

?CREATE creates a subdirectory.

STMC 2

Not used in AOS.

STMC 3

Not used in AOS.

STMC 4

Not used in AOS.

STMC 5, error, control-point-directory-name, sectors

?CREATE creates a control point directory.

STMC 6, error, filename

?CREATE creates a UDF-type file, with the user's default ACL, an element size of four, and the default number of index levels.

STMC 7, error, filename

?CREATE creates a UDF-type file, with the user's default ACL, an element size of four, and the default number of index levels.

STMC 8, error, filename

?DELETE deletes a file. If the file is a link entry, the resolution file is deleted.

STMC 9, error, directory-name

?DIR changes the working directory to directory-name. READ and EXECUTE access to directory-name are required.

STMC 10

Not used in AOS.

STMC 11, error, code

?RETURN terminates Business BASIC, passing the error code to the previous level.

STMC 12, error, programname, AC1value, AC2value

?PROCESS creates a son process and blocks the current Business BASIC process until the son terminates. Values for AC1 and AC2 must be given but they are ignored. (Zero is the recommended value for AC1 and AC2.) To use this call, you need the ability to create a son process.

STMC 13, error, programname, AC1value, AC2value

?PROCESS creates a son process without blocking the current Business BASIC process. Values for AC1 and AC2 must be given, but they are ignored. In order to use this call, you need the ability to create an unblocked son process.

STMC 14, error, flag

?PSTAT determines whether a process has sons. A 1 is returned in flag if a son process is found; otherwise, a 0 is returned.

STMC 15, error, inputconsole$

This returns @INPUT.

STMC 16, error, outputconsole$

This returns @OUTPUT.

STMC 17, error, directory$

?GNAME returns the pathname of the current directory.

STMC 18

Not used in AOS.

STMC 19

Not used in AOS.

STMC 20

Not used in AOS.

STMC 21, error, linkname, resolution-name

?CREATE creates a link entry.

STMC 22, error, string$

This causes string$ to have a length of zero.

STMC 23

Not used in AOS.

STMC 24

Not used in AOS.

STMC 25, error, oldname, newname

?RENAME renames a file.

STMC 26

Not used in AOS.

STMC 27

Not used in AOS.

STMC 28, error

?RETURN returns control to the program at the previous push level. No error code or message is returned to the previous level.

STMC 29

Not used in AOS.

STMC 30

Not used in AOS.

STMC 31

Not used in AOS.

STMC 32

Not used in AOS.

STMC 33

Not used in AOS.

STMC 34

Not used in AOS.

STMC 35, error, linkname

?GLINK verifies that linkname is a link file; if so ?DELETE deletes it.

STMC 36, error, channel

?ESFF updates the disk-resident copy of the file open on channel by flushing any modified shared pages of the file to disk.

STMC 37, error, frequency

?GHRZ returns the frequency of the system's real-time clock.

STMC 38, error, filename, mask, channel

?GOPEN opens a file for direct magnetic tape I/O.

STMC 39, error, filename, mask, channel

?OPEN opens a file for appending, for input and output, and exclusively on channel. The file must exist.

STMC 40, error, filename, mask, channel

?OPEN opens a file exclusively on channel for input and output; the file must exist.

STMC 41, error, filename, mask, channel

?OPENs a file for input only; the file must exist.

STMC 42, error, filename, mask, channel

?OPEN opens a file for input and output; the file must exist.

STMC 43, error

?BRKFL terminates the Business BASIC process and creates a break file consisting of the memory image of the terminated process. The break file is named ?pid#.time.BRK, where "pid#" is the PID of the terminated process and "time" is in the form hh_mm_ss.

STMC 44, error, filename, mask, channel

?OPEN opens a file for input and output; the file must exist.

STMC 45

Not used in AOS.

STMC 46

Not used in AOS.

STMC 47

Not used in AOS.

STMC 48

Not used in AOS.

STMC 49

Not used.

STMC 50, error, string, channel

?WRITE writes string to the file open on channel. The write is data sensitive, and the maximum length of string is 134.

STMC 51

Not used in AOS.

STMC 52, error, filename, sectors

?CREATE creates filename with an element size of sectors and zero index levels, and writes one byte at the end of the file to cause AOS to allocate space.

STMC 53

Not used in AOS.

STMEs in AOS Business BASIC

STME statements and commands allow the system manager (AA accounts) to use system calls in AOS. For more information, consult the operating system manual in which system calls are described.

Contiguous files in AOS are created by specifying an element size large enough for the entire file and zero index levels.

String variables that will receive a value should be filled with nulls before the STME is performed. You must also use the DIM (dimension) statement or command to specify the exact size stated in your operating system manual in which system calls are described. Business BASIC does not check to see if the variables you use to receive values are large enough. For this reason, if a 36-byte string is required and you supply

only a 20-byte string, the 16 bytes following the string may be appended to the string, and will overwrite part of your program. To avoid this problem, make sure you include a terminating null when you assign string arguments.

Before using any STMEs you should be thoroughly familiar with system calls.

STME 0, error, channel, pointer, filename$, template$

?GNFN, starting from pointer position, returns the next filename matching the template$ in the directory open on channel.

STME 1, error, username$

?GUNM returns the process's username.

STME 2, error, PID

?PNAME returns the process's PID.

STME 3, error, message$

?GTMES returns a CLI message.

STME 4, error, filename$, pathname$

?GNAME returns the complete pathname of filename$.

STME 5, error, buffer$, channel

?FSTAT returns the status packet for channel in the string variable buffer$.

STME 6, error, filename$, buffer$

?FSTAT returns the status of a file to buffer$.

STME 7, error, filename$, buffer$

?FSTAT returns the status of the resolution file to buffer$ when you specify a link entry.

STME 8, error, filename$, sectors, date$

?CREATE creates a contiguous file of the size specified in sectors. The time/date status area is set to the 12-byte date$.

The following illustrates the creation of a file named TESTFILE where the last accessed date and last modified date are the same as the date created.

```
0010 DIM DATE$[12]
0020 LET E=0
0030 STMA 12,E,SYS(2),SYS(1),SYS(3)-1900   :Convert the system date to Julian.
0040 LET DATE$[1,2]=CHR$(E,2)              :Date the file was created.
0050 LET DATE$[3,4]=CHR$(SYS(0)/2,2)       :Time file was created in biseconds.
0060 LET DATE$[5,8]=DATE$[1,4]             :Date & time last accesed.
0070 LET DATE$[9,12]=DATE$[1,4]            :Date & time last modified.
0080 STME 8,E,"TESTFILE<0>",4,DATE$
0090 END
```

STME 9, error, filename$, date$

?CREATE creates a file. The time/date status area is set to the 12-byte date$.

STME 10, error, pathname$, channel

?GNAM returns the complete pathname of the file open on channel.

STME 11, error, linkname$, resolution-name$

?GLINK returns the resolution name of a link.

STME 12, error, buffer$

?GCHR returns the device characteristics of @INPUT to buffer$ (6 bytes).

STME 13, error, buffer$

?GCHR returns the device characteristics of @OUTPUT to buffer$ (6 bytes).

STME 14, error, buffer$

?SCHR sets the device characteristics of @INPUT to buffer$ (6 bytes).

STME 15, error, PID, buffer$

?SEND sends the message in buffer$ to the process identified by PID.

STME 16, error, consolename$, buffer$

?SEND sends the message in buffer$ to the process controling the console.

STME 17, error, buffer$

Puts the 256-bit (32-byte) delimiter table for @INPUT into buffer$. At initialization, this table is all zero bits. This table displays the delimiters you have set with an STME 18 statement.

STME 18, error, buffer$

Sets the 256-bit (32-byte) delimiter table for @INPUT.

STME 19, error, programname$, buffer$

?PROC executes programname$, passing the contents of buffer$ to the new process as the initial IPC. The Business BASIC process is blocked until programname$ finishes.

Buffer$ must end with a <0> if data is stored in the variable. If buffer$ is zero length, and programname$ is :CLI.PR, the AOS CLI is executed much like the current implementation using STMC 12. If buffer$ contains an AOS CLI command, and :CLI.PR is in programname$, the AOS CLI command is executed, and control will be passed immediately back to the Business BASIC program after the command completes. Buffer$ may contain several AOS CLI commands separated by semicolons.

Programname$ and buffer$ must be string variables; substrings and literals are not supported.

Proper usage of STME 19 is illustrated below:

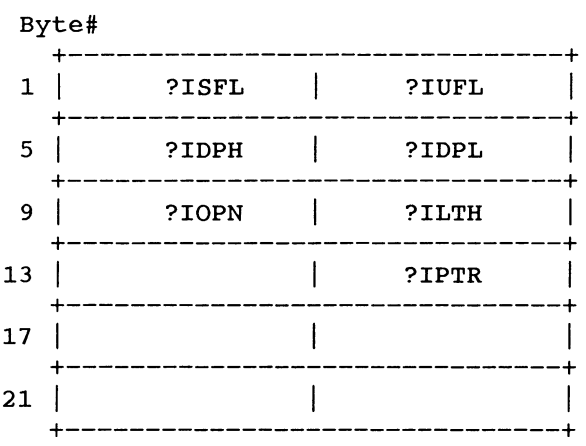
```
0030 DIM PROG$[100],BUFF$[100]
0040 LET ER=0
0050 LET PROG$=":CLI.PR<0>"
0060 LET BUFF$=""
0070 INPUT "COMMAND? ",BUFF$
0080 IF LEN(BUFF$)<>0 THEN LET BUFF$[0]="<0>"
0090 STME 19,ER,PROG$,BUFF$
```

Note that BUFF$ is null if you are PROCing the CLI to issue multiple commands; otherwise, BUFF$ contains the CLI command followed by a null.

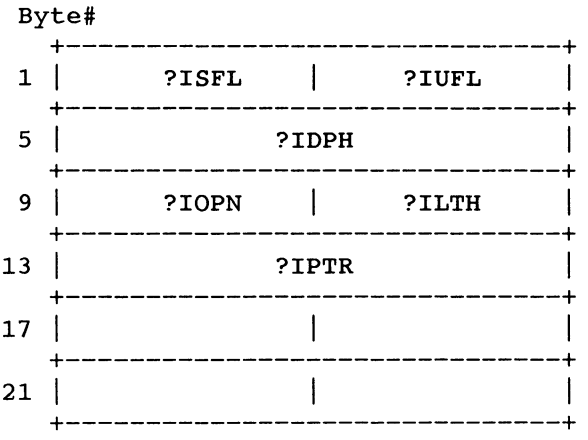
STME 20, error, control$, send$

?ISEND sends the contents of send$ via interprocess communications.

Control$ is a 24-character string. On AOS Business BASIC, control$ is formatted as follows:



On AOS/VS Business BASIC, control$ is formatted as follows:



The following flags must be set by the user:

?ISFL System Flags

?IUFL User Flags

?IDPH Destination Port, High

?IDPL Destination Port, Low

?IOPN Origin Port

The following flag may be set by the user:

?IPTR Secondary user flags if the message length is zero.

Business BASIC calculates the following:

?ILTH Message length in words. Note that the message will be rounded up to an even number of bytes.

?IPTR If the message length is nonzero, this points to the message. If the message length is zero, the user supplied value (?IPTR) is sent.

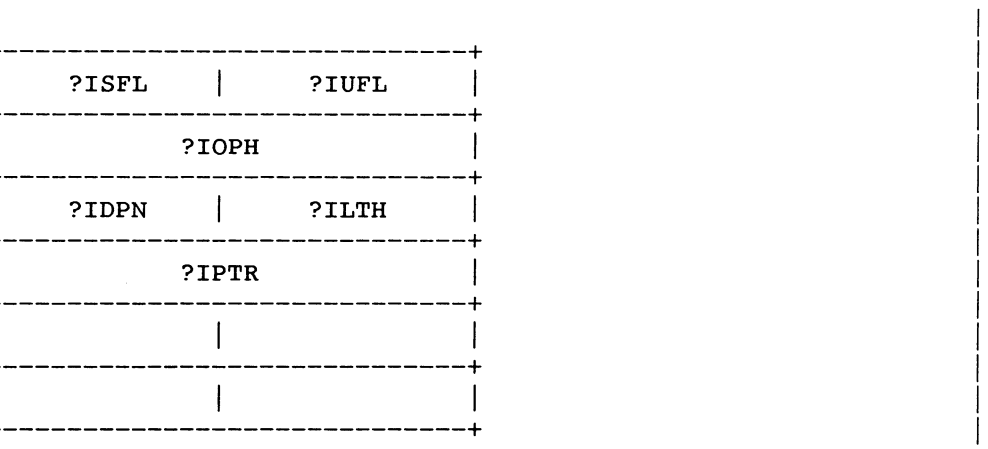
STME 21, error, control$, receive$

?IREC receives the contents of receive$ via interprocess communications.

Control$ is a 24-character string. On AOS Business BASIC, control$ is formatted as follows:

```
Byte#
    +-------------------------------+
  1 |      ?ISFL    |      ?IUFL    |
    +-------------------------------+
  5 |      ?IOPH    |      ?IOPL    |
    +-------------------------------+
  9 |      ?IDPN    |      ?ILTH    |
    +-------------------------------+
 13 |               |      ?IPTR    |
    +-------------------------------+
 17 |               |               |
    +-------------------------------+
 21 |               |               |
    +-------------------------------+
```

On AOS/VS Business BASIC, control$ is formatted as follows:



The following flags must be set by the user:

?ISFL System Flags

?IUFL User Flags

?IOPH Origin Port, High

?IOPL Origin Port, Low

?IDPN Destination Port

?ILTH The maximum message length will be set to the current word length of receive$, after receive$ has been truncated to an even number of bytes. This means that receive$ should be initialized.

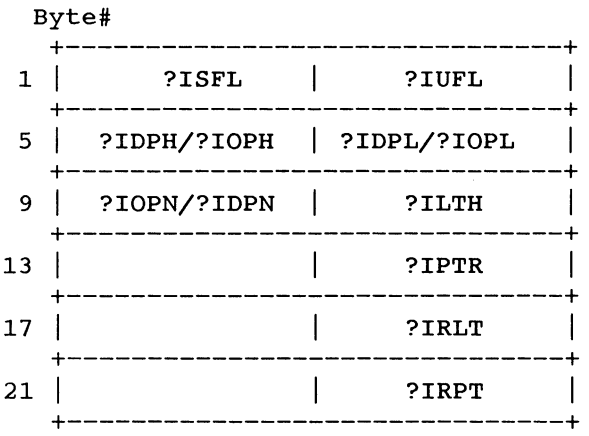
?IPTR The message pointer will be set to point to receive$. Note, however, that if a zero length message is received, Business BASIC will not update receive$ (the user may retrieve the contents of ?IPTR).

Control$ will be updated even if an error occurs.

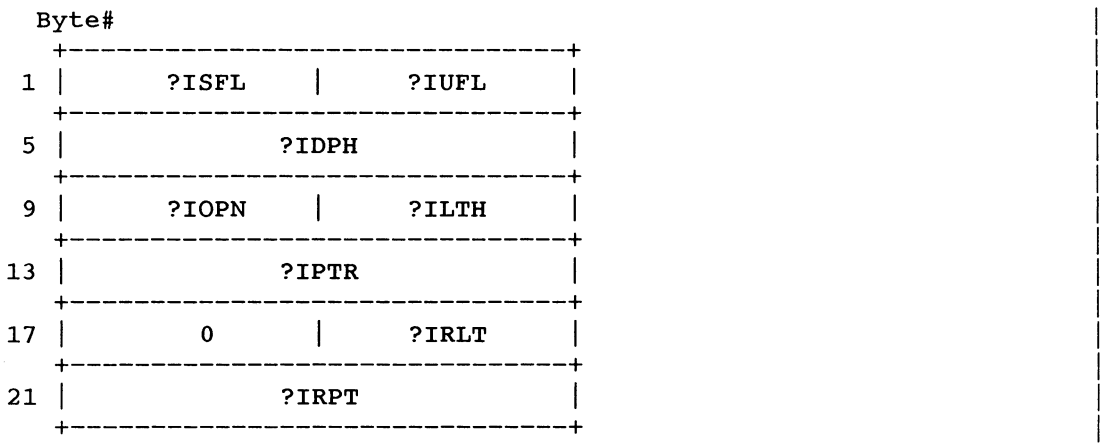
STME 22, error, control$, send$, receive$

?IS.R sends the contents of send$ via interprocess communications and then receives an interprocess communications message in receive$.

Control$ is a 24-character string. On AOS Business BASIC, control$ is formatted as follows:



On AOS/VS Business BASIC, control$ is formatted as follows:



The following flags must be set by the user:

?ISFL System Flags

?IUFL User Flags

?IDPH Destination Port, High

?IOPH Origin Port, High

?IDPL Destination Port, Low

?IOPL Origin Port, Low

?IOPN Origin Port

?IDPN Destination Port

Business BASIC performs the following functions:

?ILTH Send message length in words. Note that the message will be rounded up to an even number of bytes.

?IPTR If the message length is non-zero, this points to the message. If the message length is zero, the user-supplied value is sent.

?IRLT This maximum message length will be set to the current word length of receive$, truncated to the nearest word boundary. This means that receive$ should be initialized.

?IRPT This message pointer will be set to point to receive$.

Control$ will be updated even if an error occurs.

STME 23, error, portname$, local-port

?CREATE creates an IPC file entry with the appropriate name and local port. Note that AOS requires you to be in the directory in which Business BASIC was initially invoked in order to execute this call successfully.

STME 24, error, global-port, PID, local-port

?GPORT finds the owner of a global port. Global-port is set before the call. PID and local-port are returned.

STME 25, error, portname$, global-port

?ILKUP looks up a port number. Portname$ is set before the call. Global-port is returned.

STME 26, error, local-port, global-port

?TPORT translates a port number. Local-port is set before the call. Global-port is returned.

Remember a global port number in AOS/VS is slightly different from one under AOS. The 32-bit number is constructed as follows:

AOS/VS Bits

0 - 15 PID #

16 - 19 Ring #

20 - 31 Local port #

In AOS, the ring number does not apply; those bits are always 0.

STME 27, error, string-variable$, flag

The contents of string-variable$ are moved to an I/O buffer for display by the INPUT statement.

When you move a string with STME 27, you can use the screen edit keys to modify the contents of the string when you display it using an INPUT statement. This editing feature is available **only for terminal types 8 and 9.** String-variable$ is the string to be displayed; it must not be a literal. Flag is a number or numeric variable that controls the display of string-variable$. When flag is set to 1, string-variable$ is displayed. When flag is set to 0, string-variable$ is not displayed unless you enter Ctrl-A.

NOTE: If you press the Carriage Return key while the cursor is positioned within the string, the string is truncated at the cursor position. (You receive this same result by pressing Carriage Return while in the AOS CLI.)

RDOS Privileged System Calls

The RDOS Business BASIC privileged system calls are STMBs, STMCs, and STMDs.

For further information on devices, partitions, directories, files, links, and attributes, refer to the RDOS CLI user's manual and the RDOS system reference manual.

STMBs in RDOS Business BASIC

The STMB calls allow you to access and modify portions of memory. You must be extremely careful when you alter memory contents to be sure that you are actually modifying the locations you want. The general format of STMB statements and commands is:

STMB call, arguments

where call is the number of the STMB call and arguments vary with each call. Individual STMBs are explained on the following pages.

STMB 0, item, address

Retrieves a word address, where address is a variable receiving the word address of item. Items that are undefined return a value of 65535.

Item	Description
0	Returns the memory address of a 21-element array. Each element of this array is the address returned by the corresponding STMB 0, item, address call.
1	Returns the memory address of the line table address. The line table is an array where the first line is port 0. Each element in the table represents the word address of the start of the User Status Table for a given line. Note that detached jobs are not reflected in this array.
2	Returns the memory address of the job table address. The job table is an array. Each element in the table represents the word address of the start of the User Status Table for a given job number. Note that the number of entries corresponds with the number of jobs specified when Business BASIC was executed; i.e., an address is reserved in the User Status Table even if the job is not running.
3	Returns the memory address of a word indicating the highest multiplexor line number available to Business BASIC. This number is 2 more than the number of multiplexor lines generated into Business BASIC.

Item	Description
4	Not used.
5	Not used.
6	Returns the memory address of the channel table. The channel table is an array of N elements, where N is the number of channels specified in your BSG. Each value of N corresponds to an RDOS channel number. The value contained in an array element is either: ● the User Status Table address of the user who has a file opened on that channel, or ● a Business BASIC system address, if Business BASIC has opened a file on that channel. This information corresponds to the first two columns of information returned by the utility program SCHANS.
7	Returns the memory address of the reserved file table. This table contains the reserved file names generated by the options chosen in the BSG program (for example, $LPT) and any additional reserved file names specified in the BSG questions.
8	Returns the memory address of a word containing the maximum amount of memory available for the current job.
9	Returns the memory address of a two-element array that represents the global switches on Business BASIC.
10	Returns the memory address of a word containing the number of jobs specified when Business BASIC was executed.
11	Not used.

Item	Description
12	Returns the memory address of the lock buffer area, which is an array of 11 by N elements, where N is the number of locks specified in the BSG. Words in each lock buffer have the following meanings: 1 Reserved 2 Reserved 3 User job number 4 Lock number assigned by user 5 Filename (10 characters) 6 Filename (10 characters) 7 Filename (10 characters) 8 Filename (10 characters) 9 Filename (10 characters) 10 Starting sector number 11 Ending sector number
13	Not used.
14	Returns the memory address of a word that indicates the length in blocks of one segment of the push file (BASIC.PS). This number is determined by the following formula: segment length = (maximum-program-size + 1) * (number-of-jobs) where the maximum program size is determined upon execution with the /M switch, allocated in 512 byte blocks, and the number of jobs is specified at execution with the /J switch.
15	Returns the memory address of a double-word that is used by SYS(29).
16	Not used.
17	Not used.
18	Not used.

Item	Description
19	Returns the memory address of a word containing the address of the current user's User Status Table.
20	Not used.
21	Returns the address of the physical channel number for the push file (BASIC.PS).

STMB 1, wordcount, address, destination

Copies the contents of memory starting at address (word address) for the number of words specified in wordcount to the variable or array destination.

STMB 2, wordcount, address, source

Copies the specified number of words in wordcount from the variable or array source into memory at the address (word address) specified.

STMB 3, byteaddress, destination$

Copies bytes from memory beginning at byteaddress (word address * 2) into the string variable or substring destination$ until destination$ is full.

STMB 4, byteaddress, source$

Copies the entire contents of string variable or substring source$ into memory starting at the address specified in byteaddress.

STMB 5

Not used.

STMB 6, job-number, error

Attaches the specified detached job to the current terminal, returning a possible error in variable error. Note also that the job which was attached to the current terminal (and which issued the STMB 6 call) becomes detached. Errors you may receive are:

51 Job not logged on
70 Invalid job number
71 Job already attached

STMB 7, job-number, error

Forces the specified job to execute a BYE command, returning a possible error in the variable error. Errors you may receive are:

51 Job not logged on
70 Invalid job number
78 MSG in progress

STMB 8, job-number, error

Resets the ON IKEY condition for the specified job, clears the "ignore IKEY" flag, and then simulates an interrupt, stopping the program. Errors you may receive are:

51 Job not logged on
70 Invalid job number

STMB 9

Not used.

STMB 10, function, address, mask

If function = 1, STMB 10 sets the bits in the word at the specified word address according to the bits set in the variable mask. If function = 0, STMB 10 clears the bits in the word at the specified address according to the bits set in the variable mask.

STMB 11, string$, variable1, variable2, variable3

Scans the contents of string$, puts the number of 1 bits in variable1, and puts the largest consecutive number of 0 bits in variable2. Variable3 is required and is set to 0. String$ may be a substring, but it must be word-aligned and of even byte length—that is, the subscripts of string$[x,y] must be such that x is odd and y is even.

STMB 12, job-number

Scans the job table for the first available job and executes HELLO for that job. The system returns the number of the job in the variable job-number. If no available job exists, it returns a -1 in job-number. Note that when this call finds an available job, it starts the job as a detached job. The HELLO program waits 15 seconds for input and if none is received, it does a BYE for that job.

STMB 13, job-number, error, flag, string

Places the contents of string into the input buffer for the specified job. The system

sets the alternate IKEY or unpend flag value for job-number to the value of flag (0 or 1), and unpends the job. Errors you can receive are:

51 Job not logged on
70 Invalid job number
72 Job is not waiting on input

If you specify any job that is running (other than your own job), you get error = 0 (successful), unless input for that job is currently being processed, in which case you get error 72. If the job to which you send a string is waiting on an INPUT statement, but you send it an inappropriate string, it displays a prompt for more input (\?). If the job to which you send a string is at the asterisk prompt, but you send it an inappropriate command, you will receive the error message `ERROR 2 - STATEMENT OR COMMAND SYNTAX IS INVALID`.

If operating system multiplexor support is chosen, using STMB 13 in a tight loop to send commands to a job can cause a system hang, due to timing problems associated with the .OPEN and .CLOSE system calls. To avoid this problem, you can add a DELAY 0 statement after each STMB 13.

The STMB 13 call can be used in conjunction with the STMB 12 call to start up a job, if there is one available. STMB 13 would be used to supply HELLO with answers to the account name, password, and directory/program questions.

STMB 14, job-number, error, flag

Sets the alternate IKEY or unpend flag value for job-number to the value of flag (0 or 1), and the job acts as if an interrupt occurred. Errors you can receive are:

51 Job not logged on
70 Invalid job number

If you specify any job that is running (other than your own job), regardless of whether it is running a program or sitting at the Business BASIC prompt, you get error = 0 (successful).

STMB 15, port-number, characteristics

Sets the multiplexor line to the specified line characteristics. Port number 0 corresponds to multiplexor line 2. If characteristics is set to -1, then the system returns the value to which the line characteristics was last set. For invalid lines, characteristics is set to -1.

The line characteristics word contains the following:

```
Bit(s)   0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1
         0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
```

Content - x x x x x C C C C S S D D P P x (ALM, ULM)

Content - x x x x C C C C S S P P D D x x (ASLM, USAM)

where:

CCCC = Clock number

Baud rates corresponding to clock numbers are:

ALM: depend on how the board is jumpered
ULM:

0	0	4	134.5	8	9600	12	2400
1	19200	5	200	9	4800	13	300
2	50	6	600	10	1800	14	150
3	75	7	2400	11	1200	15	110

ASLM, USAM:

0	50	4	150	8	1800	12	4800
1	75	5	300	9	2000	13	7200
2	110	6	600	10	2400	14	9600
3	134.5	7	1200	11	3600	15	19200

SS= Number of stop bits

ALM, ULM:
- 00 = 1 stop bit
- 01 = 2 stop bits

ASLM, USAM:
- 01 = 1 stop bit
- 10 = 1 stop bit
- 11 = 2 stop bits

DD= Number of data bits
- 00 = 5 data bits
- 01 = 6 data bits
- 10 = 7 data bits
- 11 = 8 data bits

PP= Parity type

ALM,ULM:
- 00 = none
- 01 = odd
- 10 = even
- 11 = mark

ASLM, USAM:
00 = odd, disabled
01 = odd, enabled
10 = even, disabled
11 = even, enabled

x = Meaningless. The interpreter will disregard these bits and will always return them to the user set to 0.

STMB 16, flag

Sets (flag = 1) or clears (flag = 0) the run-only flag in the current program. If set, this flag prevents a file from being listed or modified by non-AA accounts. For information about run-only programs, see Chapter 5.

STMB 17

STMB 17 executes the .RTN system call, which shuts down Business BASIC. If you are using Business BASIC multiplexor support, STMB 17 also shuts down the multiplexor.

STMB 18, address, value

Returns in value the word stored at address in the operating system's address space.

STMB 19, port-number, modem-status

Sets the multiplexor line to the modem status. Port number 2 corresponds to multiplexor line 0. If status is set to -1, then the system returns the value to which the modem status was last set. For invalid lines, status is set to -1.

Note that the interpreter manipulates Request To Send and Data Terminal Ready when input and output are performed. Thus, you cannot write a Business BASIC program to drive a half-duplex device. STMB 19 is used to reinitialize modem lines after the abnormal termination of a job logged on the line.

The modem status word contains the following:

```
Bit(s)   0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1
         0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
```

Content - x x x x x x x x x x x x x x R D (ALM, ULM)

Content - x x x x x x x x x x R x x x D x (ASLM, USAM)

where:

R = Request To Send
D = Data Terminal Ready
x = Meaningless. The interpreter will disregard these bits and will always return them to the user set to 0.

STMB 20

Performs an immediate BYE to log off the current job, completes system housekeeping, and returns to the log-on banner.

STMB 21

Not used.

STMB 22

Not used in RDOS.

STMB 23

Not used in RDOS.

STMB 24

Not used in RDOS.

STMB 25, user-channel, system-channel

Maps the user channel to the system channel, making the system channel accessible to the user. This is used to access the push file, BASIC.PS, in utilities like RNAM, VAR, and PD. A CLOSE on a user channel opened with STMB 25 does not close the physical channel but frees the user channel.

STMCs in RDOS Business BASIC

The STMC statement or command translates directly into the RDOS system call described with it. Refer to the RDOS system reference manual for descriptions of the system calls, the arguments needed, and the errors returned. Business BASIC checks for compatibility of variable types; it does not check to see if the variables you use to receive values are large enough. Therefore, if a 36-byte string is required and you supply only a 20-byte string, the 16 bytes following the string will overwrite part of your program.

When you use a string literal as an argument, the literal must include a terminating null. Without the null, the operating system will read beyond the end of the literal and

produce erroneous results. In order to pass the literal "ABC", you must write the literal as ABC<0>.

When you use a string variable as an argument to receive a value, fill the variable with nulls before using the STMC.

Arguments that refer to channel numbers may refer to absolute operating system channels or to program-relative BASIC channels. A negative channel number refers to the operating system channel; e.g., -6 refers to operating system channel 6. A positive channel number refers to a file opened in the current BASIC program on this relative channel; e.g., 2 refers to the file opened in the current program on channel 2. Therefore, channel numbers for STMCs range from -255 to 15. Operating system channel 0 is a special case that is passed as -0 (32768). You should use absolute channel numbers only to refer to Business BASIC channel numbers.

Business BASIC leaves a user and/or system channel allocated when an STMC attempts to open a file and fails. Therefore, if an error occurs, you should close the channel by using STMC 53 if the channel is negative (i.e., a system channel) or by using CLOSE FILE(channel) if the channel is positive (i.e., a Business BASIC user channel).

When a user accesses a file through the standard BASIC statements (OPEN, READ, PRINT, CLOSE, etc.), the interpreter inserts directory specifiers before the user-specified filename to indicate the user's current directory, which is stored in the User Status Table, or the library directory (e.g., $LIB). This is not necessarily the same as the system directory, which is where RDOS will look for files that are being created or opened with STMCs when no directory is specified. Thus, a directory should always be specified explicitly when STMCs are used.

Users may explicitly supply directory specifiers only if a special bit in the User Status Table is set (see Appendix A); otherwise, any filename containing a colon is rejected unless it is in the Reserved Filename table. STMCs and reserved filenames are not handled in this manner; if no directory is specified by the user, no directory is specified by the interpreter. Thus, the operating system will look for the file in the current system directory, unless it is a file that is treated specially by the operating system. Since it is possible to change the current system directory at runtime (!SDIR, DIR, STMC 9), links to reserved files and other special files which are accessed using STMCs must be handled very carefully.

The general format for STMC is:

```
STMC call, error, arguments
```

where call is one of the following calls (by number); error is a variable to receive the RDOS error returned (a positive error code is returned, so negate it to find it in the BASIC.ER file; -1 is returned if the call is successful); and arguments vary with each call.

The individual STMCs are explained on the following pages.

STMC 0, error, filename, sectors

.CCONT creates a contiguous file with all locations initialized to 0.

STMC 1, error, directory-name

.CDIR creates a subdirectory.

STMC 2, error, attributes, channel

.CHATR changes the attributes for the file opened on channel.

STMC 3, error, attributes, channel

.CHLAT changes the link attributes of the link file opened on channel.

STMC 4, error, buffer$, channel

.CHSTS returns the 36-byte status table for the file opened on channel.

STMC 5, error, partitionname, sectors

.CPART creates a subpartition.

STMC 6, error, filename

.CRAND creates a random file.

STMC 7, error, filename

.CREAT creates a sequential file.

STMC 8, error, filename

.DELET deletes a file. If the file is a link entry, the resolution file will be deleted.

STMC 9, error, directory-name

.DIR changes the current system directory. This does not change the default user directory, which is stored in the User Status Table.

STMC 10, error, oldname, newname

.EQIV renames a device.

STMC 11, error, code

.ERTN terminates Business BASIC, passing the error code to the previous level.

STMC 12, error, programname, AC1value, AC2value

.EXBG puts a checkpoint in the background and executes the program you specify. Business BASIC itself cannot have checkpoints.

STMC 13, error, programname, AC1value, AC2value

.EXFG executes programname in the foreground partition. An AC2value must be given but it is ignored.

STMC 14, error, flag

.FGND determines whether a foreground program is already running. Flag = 1 is returned if a foreground program is found; otherwise, flag = 0.

STMC 15, error, inputconsole$

.GCIN returns the name of the console input device for this ground.

STMC 16, error, outputconsole$

.GCOUT returns the name of the console output device for this ground.

STMC 17, error, directory$

.GDIR returns the name of the current Business BASIC system directory.

STMC 18, error, attributes, channel

.GTATR returns the attributes of the file opened on channel.

STMC 19, error, system$

.GSYS returns the current operating system's name.

STMC 20, error, devicename, flag

.INIT initializes a device into the system. Flag = -1 specifies full initialization; flag <> -1 specifies partial initialization.

STMC 21, error, linkname, resolution-name

.LINK creates a link entry.

STMC 22, error, masterdirectory$

.MDIR returns the name of the current master directory.

STMC 23, error

.ODIS disables console keyboard interrupts.

STMC 24, error

.OEBL enables console keyboard interrupts.

STMC 25, error, oldname, newname

.RENAM renames a file.

STMC 26, error

.RESET closes all Business BASIC system and user files currently opened in this ground. This includes the BASIC.PS and BASIC.ER files and the user and system program libraries.

STMC 27, error, devicename

.RLSE releases a previously initialized device.

STMC 28, error

.RTN returns control to the program at the previous push level.

STMC 29, error, day, month, year-1968

.SDAY sets the current system date.

STMC 30, error, devicename

.SPDA disables spooling for a device.

STMC 31, error, devicename

.SPEA enables spooling for a device.

STMC 32, error, devicename

.SPKL kills spooling for a device.

STMC 33, error, filename, buffer$

.STAT returns the 18-byte status of the file you specify.

STMC 34, error, seconds, minutes, hours

.STOD sets the time of day.

STMC 35, error, linkname

.ULNK removes a link entry.

STMC 36, error, channel

.UPDAT updates the disk-resident copy of the file opened on channel.

STMC 37, error, frequency

.GHRZ returns the frequency of the system's real-time clock.

STMC 38, error, filename, mask, channel

.MTOPD opens a file for direct magnetic tape I/O.

STMC 39, error, filename, mask, channel

.APPEND opens a file for appending.

STMC 40, error, filename, mask, channel

.EOPEN opens a file for exclusive use.

STMC 41, error, filename, mask, channel

.ROPEN opens a file for read-only access.

STMC 42, error, filename, mask, channel

.OPEN opens a file for shared access.

STMC 43, error

.BREAK terminates Business BASIC and saves the status in BREAK.SV.

STMC 44, error, filename, mask, channel

.TOPEN transparently (without changing the date last written or accessed) opens a file you specify for exclusive access.

STMC 45, error, filename, sectors, date$

.TCCONT transparently creates a contiguous file. The time/date status area is set to the 6-byte date$.

The format of date$ for STMCs 45, 46, and 47 is:

- bytes 1-2 Julian value, date last accessed
- bytes 3-4 Julian value, creation date
- bytes 5-6 Creation time, one-half of seconds past midnight

STMC 46, error, filename, date$

.TCRND transparently creates a random file. The time/date status area is set to the 6-byte date$.

STMC 47, error, filename, date$

.TCRET transparently creates a sequential file. The time/date status area is set to the 6-byte date$.

STMC 48, error, AC0

.GMEM gets the current memory allocation for the current ground.

STMC 49

Not used.

STMC 50, error, string, channel

.WRL writes a line to the file opened on channel.

STMC 51, error, filename, buffer$

.RSTAT returns the status of the resolution file when you specify a link entry.

STMC 52, error, filename, sectors

.CONN creates a contiguous file without initializing the file to nulls.

STMC 53, error, channel

.CLOSE closes the file opened on channel. Channel must be negative; use CLOSE FILE for positive channels.

STMDs in RDOS Business BASIC

STMD statements and commands allow the system manager (an AA account) to send messages to and receive responses from any user's terminal.

The STMD statements and commands send the message string literally, so if you need carriage returns or control characters, you have to enclose their ASCII decimal values in angle brackets. For example, <13> causes a carriage return or line feed to occur at the position in the message string where it appears. Output of the message string stops on a null or end of string, whichever comes first.

The STMD statements and commands are:

STMD 0, port-number, 0, message$

STMD 0, port-number, wait, message$, reply$

STMD 1, port-number, 0, message$

STMD 1, port-number, wait, message$, reply$

STMD 1 overrides any no-message flag set by the terminal; STMD 0 does not. You must supply the port number and your message.

You can supply either a 0 for no wait time and no reply, or a numeric expression for wait and a string variable reply$ to receive a reply. If you specify wait and supply a string variable reply$, the system waits for a reply according to the rules of TINPUT USING. Each character received into reply$ updates the current string length. If you don't get a reply in the specified amount of time, the system executes the next statement in your program (or returns you to keyboard mode) without changing the contents of reply$. You can then check SYS(22) to see if a timeout occurred.

You cannot request a reply from a terminal that is not logged on to Business BASIC if you are using operating system multiplexor support. However, if the terminal is logged on, you can request STMD with reply.

During the execution of an STMD, it is as if the job that issued the call had attached to the destination terminal. Interrupts are disabled for the job issuing the call. When no reply is requested, any characters struck at the destination terminal while the message is being displayed will be echoed later at the requesting job's terminal, and any input at the requesting terminal will be discarded.

The example below shows an STMD that forces a message to terminal number 1. The bell character (<7>) is included, along with a carriage return (<13>). The STMD waits 30 seconds for a reply; if a timeout occurs, line 110 directs control back to line 100. The wait flag is in tenths of seconds.

```
0100 STMD 1,1,300,"<13>Tell me your name in 30 seconds!<7>",REPLY$
0110 IF SYS(22)=0 THEN GOTO 0100
0120 PRINT REPLY$
```

End of Chapter

Appendix A
User Status Tables

In AOS and RDOS Business BASIC, each job maintains a User Status Table containing information about the job. For AOS or RDOS, you can retrieve the address of the User Status Table for the current job by executing STMB 1,1,14,address. In this case, address receives the word address of the start of the table. For AOS/VS and AOS/VS II, you can use STMB 5,2,14,address to retrieve the address of the User Status Table for the current job. Address receives the double-word address of the start of the table.

The STMA statements and commands change these tables. Some programs (HELLO, STAT, KILL, OPCLI) require direct access to this information. Under AOS, you probably never need this information, since the operating system handles all system tuning and resources. Many of the table locations are used only in RDOS Business BASIC. If you modify data in the user tables incorrectly, you could cause a system crash.

The User Status Tables are subject to change. Refer to the file USERSTATUS in $DOC for the documentation of the current user tables. The structure of the user status table is contained in the BASICGEN directory in either PARBU.SR (RDOS and AOS) or PS_PARBU.SR (AOS/VS and AOS/VS II).

End of Appendix

Appendix B
ASCII Character Set

Keys	Definition	Octal	Decimal	Hexadecimal
^@	Null	000	000	000
^A	Print form	001	001	001
^B		002	002	002
^C	Enable blink	003	003	003
^D	Disable blink	004	004	004
^E	Read cursor addr	005	005	005
^F		006	006	006
^G	Bell	007	007	007
^H	Cursor home (HOME)	010	008	008
^I	Tab (TAB)	011	009	009
^J	New line (NEW LINE)	012	010	00A
^K	Erase to EOL (ERASE EOL)	013	011	00B
^L	Erase page (ERASE PAGE)	014	012	00C
^M	Carriage return (CR)	015	013	00D
^N	Start blink	016	014	00E
^O	End blink	017	015	00F
^P	Position cursor	020	016	010
^Q	Print	021	017	011
^R	Roll enable	022	018	012
^S	Roll disable	023	019	013
^T	Start underscore	024	020	014
^U	End underscore	025	021	015
^V		026	022	016
^W	Cursor up	027	023	017
^X	Cursor right	030	024	018
^Y	Cursor left	031	025	019
^Z	Cursor down	032	026	01A
^[	Escape (ESC)	033	027	01B
^\	Start dim	034	028	01C
^]	End dim	035	029	01D
^^	Function key prefix	036	030	01E
^_	Cursor addr response	037	031	01F

ASCII Character Set

Key	Octal	Decimal	Hexadecimal	Key	Octal	Decimal	Hexadecimal
space bar	040	032	020	P	120	080	050
!	041	033	021	Q	121	081	051
"	042	034	022	R	122	082	052
#	043	035	023	S	123	083	053
$	044	036	024	T	124	084	054
%	045	037	025	U	125	085	055
&	046	038	026	V	126	086	056
'	047	039	027	W	127	087	057
(	050	040	028	X	130	088	058
)	051	041	029	Y	131	089	059
*	052	042	02A	Z	132	090	05A
+	053	043	02B	[	133	091	05B
,	054	044	02C		134	092	05C
-	055	045	02D	]	135	093	05D
/	057	047	02F	_	137	095	05F
0	060	048	030	`	140	096	060
1	061	049	031	a	141	097	061
2	062	050	032	b	142	098	062
3	063	051	033	c	143	099	063
4	064	052	034	d	144	100	064
5	065	053	035	e	145	101	065
6	066	054	036	f	146	102	066
7	067	055	037	g	147	103	067
8	070	056	038	h	150	104	068
9	071	057	039	i	151	105	069
:	072	058	03A	j	152	106	06A
;	073	059	03B	k	153	107	06B
<	074	060	03C	l	154	108	06C
=	075	061	03D	m	155	109	06D
>	076	062	03E	n	156	110	06E
?	077	063	03F	o	157	111	06F
@	100	064	040	p	160	112	070
A	101	065	041	q	161	113	071
B	102	066	042	r	162	114	072
C	103	067	043	s	163	115	073
D	104	068	044	t	164	116	074
E	105	069	045	u	165	117	075
F	106	070	046	v	166	118	076
G	107	071	047	w	167	119	077
H	110	072	048	x	170	120	078
I	111	073	049	y	171	121	079
J	112	074	04A	z	172	122	07A
K	113	075	04B	{	173	123	07B
L	114	076	04C	|	174	124	07C
M	115	077	04D	}	175	125	07D
N	116	078	04E	~	176	126	07E
O	117	079	04F	DEL	177	127	07F

End of Appendix

Appendix C
Function Key Character Set

Function Key (F)	(F)+Shift	(F)+Shift+Ctrl	(F) only	(F)+Ctrl
ERASE PAGE	012	12	012	12
PRINT*				
ERASE EOL	011	11	011	11
TAB	009	09	009	09
C1	30 088	30 88	30 092	30 92
C2	30 089	30 89	30 093	30 93
C3	30 090	30 90	30 094	30 94
C4	30 091	30 91	30 095	30 95
F1	30 097	30 33	30 113	30 49
F2	30 098	30 34	30 114	30 50
F3	30 099	30 35	30 115	30 51
F4	30 100	30 36	30 116	30 52
F5	30 101	30 37	30 117	30 53
F6	30 102	30 38	30 118	30 54
F7	30 103	3030 119	30 55	
F8	30 104	30 40	30 120	30 56
F9	30 105	30 41	30 121	30 57
F10	30 106	30 42	30 122	30 58
F11	30 107	30 43	30 123	30 59
F12	30 108	30 44	30 124	30 60
F13	30 109	30 45	30 125	30 61
F14	30 110	30 46	30 126	30 62
F15	30 096	30 32	30 112	30 48
UP	023	23	023	23
RIGHT	024	24	024	24
LEFT	025	25	025	25
DOWN	026	26	026	26
HOME	008	08	008	08

*No value.

End of Appendix

Appendix D
Error Messages

This appendix lists all the error messages that Business BASIC returns. Under AOS, your system can also return exceptional condition messages described in the AOS CLI user's manual and the AOS programmer's manual. RDOS exceptional condition messages are included in the Business BASIC error file. The first number shown in these tables is the record number in the BASIC.ER file of error messages. The second number is the error code that Business BASIC returns in SYS(7). You can use either number in the ERM$ string function. You can use File Maintenance and the table file ERRORS.TB to add your own error messages at the end of the file in the range 640-1023.

Unless a specific section indicates otherwise, these tables use "AOS" to refer to AOS, AOS/WS, AOS/VS, and AOS/VS II. When differences exist, "AOS/VS" and "32-bit only" are used to refer to AOS/VS and AOS/VS II (the 32-bit operating systems), and "AOS" and "16-bit only" are used to refer to AOS and AOS/WS (the 16-bit operating systems). Similarly, "RDOS" refers to RDOS and DG/RDOS unless otherwise noted.

Table D-1 Business BASIC Error Messages

Record Number in BASIC.ER	Error Code Returned in SYS(7)	Error Message
1	1	Illegal character
2	2	Statement or command syntax is invalid
3	3	READ/DATA types inconsistent
4	4	SYSTEM (program is lost)
5	5	Statement number
6	6	Excessive number of variables (more than 348)
7	7	Statement may not be used as a command
8	8	Specification
9	9	Illegal reserved file name
10	10	Reserved file in use
11	11	Parenthesis
12	12	Illegal command
13	13	Line number
14	14	No more program memory
15	15	End of DATA

Table D-1 Business BASIC Error Messages *(continued)*

Record Number in BASIC.ER	Error Code Returned in SYS(7)	Error Message
16	16	Arithmetic
17	17	Unassigned variable
18	18	GOSUB nesting
19	19	RETURN - no GOSUB
20	20	FOR nesting
21	21	FOR - no NEXT
22	22	NEXT - no FOR
23	23	No more data memory
24	24	No available channels
25	25	Option is not in this system
26	26	Program/data overflow
27	27	Illegal file number
28	28	New dimension exceeds old dimension
29	29	Expression
30	30	Illegal mode
31	31	Subscript
32	32	Undefined function
33	33	Function nesting
34	34	Function argument
35	35	Illegal format string
36	36	String size
37	37	User routine
38	38	Undimensioned string
39	39	Duplicate matrix
40	40	Matrices sizes
41	41	Matrix DIM
42	42	File already opened
43	43	Matrix not square
44	44	File unopened
45	45	Illegal record length
46	46	Input invalid
47	47	Wrong mode
48	48	Not a save file
49	49	No room for directory
50	50	Invalid operator command
51	51	User not on system
52	52	User in NOMSG state
53	53	Renumbering error(s)
54	54	Statement length
55	55	Block I/O error
56	56	Attempt to LOCK same area twice
57	57	Database record is LOCKed
58	58	Incompatible save file

Table D-1 Business BASIC Error Messages *(concluded)*

Record Number in BASIC.ER	Error Code Returned in SYS(7)	Error Message
59	59	Zero STEP
60	60	Line too long
61	61	Missing right parenthesis in format
62	62	Save file is run only
63	63	Incorrect number of arguments
64	64	Incorrect argument type
65	65	Program swapping error
66	66	String argument in error
67	67	Error in index file or index parameters
68	68	Index file full
69	69	Excessive IPB errors
70	70	Invalid job/terminal number
71	71	Job/terminal already attached
72	72	Job is not waiting on input
73	73	Edit buffer is empty
75	75	Program may not be listed
76	76	Non-fatal system error
77	77	Illegal record number
78	78	MSG in progress
79	79	Undefined RLS2 function code
80	80	Error executing ON ERR statement
81	81	Illegal record status
82	82	Data file full
83	83	Error from INFOS II
84	84	Not a database file
85	85	Illegal command with optimized file
86	86	LFTABL$ string error
87	87	Missing ELSE or END IF
88	88	Extra ELSE or missing END IF
89	89	Illegal file type
90	90	Logical file does not exist
91	91	Attempt to issue LOCK/UNLOCK without RLS2 running
92	92	Parameter range error
93	93	Maximum number of locks has been exceeded
94	94	Incompatible revision of volume label file

Table D-2 Utility Program Error Messages

Record Number in BASIC.ER	Error Code Returned in SYS(7)	Error Message
128	128	Invalid or out-of-range field
129	129	No key was given
130	130	Record does not exist
131	131	Record already exists
132	132	Validation error, key was changed
133	133	Function is invalid without a FIND function first
134	134	Control file is in error
135	135	Invalid format or page
136	136	No next record
137	137	Index for this record may be all messed up
138	138	Key could not be deleted
139	139	Request not completed
140	140	Invalid account/password
141	141	INPUT timed out
142	142	Device is assigned
143	143	Illegal function
144	144	File not found
146	146	Key already exists and duplicates not allowed
147	147	Insufficient space for logical file
148	148	File not on sector boundary
149	149	Record out of sequence
150	150	Illegal blocking factor
151	151	Illegal key length
152	152	Data dictionary does not exist
153	153	File types do not match
154	154	Wrong program

Table D-3 RDOS System Error Messages

Record Number in BASIC.ER	Error Code Returned in SYS(7)	Error Message
256	0	Illegal channel number
257	-1	Illegal file name
258	-2	Illegal system command
259	-3	Illegal command for device
260	-4	Not a save file
261	-5	File already exists
262	-6	End of file
263	-7	File read protected
264	-8	File write protected
265	-9	File already exists
266	-10	File does not exist
267	-11	Permanent file
268	-12	File attribute protected
269	-13	File not open
270	-14	Fatal utility error
271	-15	Execute CLI.CM (no error)
272	-16	Invisible error code
273	-17	Channel already in use
274	-18	Line too long
275	-19	Attempt to restore a non-existent image
276	-20	Parity error
277	-21	Push depth exceeded
278	-22	Insufficient memory to execute program
279	-23	File space exhausted
280	-24	File data error
281	-25	Unit improperly selected
282	-26	Illegal starting address
283	-27	Attempt to read into system space
284	-28	File accessable by block I/O only
285	-29	Files on different directories
286	-30	Device not in system
287	-31	Illegal overlay number
288	-32	File not accessable by block I/O
289	-33	Invalid time or date
290	-34	Out of TCBs
291	-35	Signal to busy address
292	-36	File already squashed error
293	-37	Device already in system
294	-38	Insufficient contiguous blocks
295	-39	Simultaneous read or write to mux line
296	-40	Error in user task queue table
297	-41	No room for directory
298	-42	Illegal directory specifier
299	-43	Unknown directory specifier

Table D-3 RDOS System Error Messages *(continued)*

Record Number in BASIC.ER	Error Code Returned in SYS(7)	Error Message
300	-44	Partition too small
301	-45	Directory depth exceeded
302	-46	Directory in use
303	-47	Link depth exceeded
304	-48	File in use
305	-49	Task ID error
306	-50	Common size error
307	-51	Common usage error
308	-52	File position error
309	-53	Insufficient room in data channel map
310	-54	Directory not initialized
311	-55	No default directory
312	-56	Foreground already active
313	-57	Error in partition set
314	-58	Directory shared
315	-59	No room for UFTs
316	-60	Address error on .SYSTM argument (**NOTE:** Use the AERM$ function to retrieve the operating system error.)
317	-61	Not a link entry
318	-62	Program not checkpointable
319	-63	SYS.DR error
320	-64	MAP.DR error
321	-65	Device timeout
322	-66	Entry not accessable by a link
323	-67	Not a source file
324	-68	Transmission terminated by receiver
325	-69	System deadlock
326	-70	I/O terminated by channel close
327	-71	Spool files active
328	-72	Task not found for abort
329	-73	Device previously opened
330	-74	System stack overflow
331	-75	No MCA receive request outstanding
332	-76	Attempt to release an open device
333	-77	.XMT or .IXMT messages must be non-zero
334	-78	'You can't do that'
335	-79	.TOVLD not loaded for queued overlay tasks
336	-80	Operator messages not SYSGENed
337	-81	Disk format error
338	-82	Disk has invalid bad block table
339	-83	Insufficient space in bad block pool (core)

Table D-3 RDOS System Error Messages *(concluded)*

Record Number in BASIC.ER	Error Code Returned in SYS(7)	Error Message
340	-84	Attempt to create a zero length contiguous file
341	-85	Program is not swappable
342	-86	Blank tape
343	-87	ALM line not ready
344	-88	Console interrupt received
345	-89	Read overrun error
346	-90	Read framing error
347	-91	Too many soft errors
348	-92	QTY buffer overflow

Table D-4 Business BASIC I/O Error Messages

Record Number in BASIC.ER	Error Code Returned in SYS(7)	Error Message
383	-127	Device is assigned, but not to you
384	-128	Illegal function
385	-129	Variable length blocks not allowed
386	-130	Rewrite mode not allowed for non-disk device
387	-131	Illegal function for device
388	-132	Open processing error
389	-133	Function does not process specified record format
390	-134	File already in use
391	-135	File currently opened for exclusive use
392	-136	File not open
393	-137	Peripheral device currently opened
394	-138	VL file processing error
395	-139	Unresolved resource conflict
396	-140	Rewrite inverted with changed data address
397	-141	File name already in use
398	-142	Block size exceeds window
399	-143	Virtual memory exhausted
400	-144	System translation table load error
401	-145	The system cannot open the file as requested
402	-146	Volume close error
403	-147	Insufficient space to open file
404	-148	Beyond logical bounds of file
405	-149	Translation specification error
406	-150	Volume does not exist
407	-151	Record is locked and 'hold' not requested
408	-152	Disk space exhausted
409	-153	Record beyond bounds of file
410	-154	Error while closing volume during internal volume switch
411	-155	Physical I/O error
412	-156	Residual disk error
413	-157	Disk or magnetic tape timeout
414	-158	Illegal access method
415	-159	Insufficient parameters for translation
416	-160	Special file open error in preopen
417	-161	Special file close error in preopen
418	-162	Internal special file close error
419	-163	RDOS open failure
420	-164	Volume already exists
421	-165	Zero length transfer request on disk device
422	-166	ISAM update conflict
423	-167	Index naming error
424	-168	Index not in database index definition file

Table D-4 Business BASIC I/O Error Messages *(continued)*

Record Number in BASIC.ER	Error Code Returned in SYS(7)	Error Message
425	-169	File name too long
426	-170	No node space available
427	-171	Mag tape processing error
428	-172	System does not support device
429	-173	End of volume for output file
430	-174	End of volume for input file
431	-175	ISAM comparison error
432	-176	ISAM resolution error
433	-177	Illegal relative motion
434	-178	Invalid node address encountered internally
435	-179	Invalid entry address encountered internally
436	-180	Top level error
437	-181	Subindices not allowed
438	-182	Subindex not present
439	-183	(sub)index boundary encountered
440	-184	Delete positioning error
441	-185	Multiple key write error
442	-186	Key too long or of zero length
443	-187	Invalid entry number encountered internally
444	-188	Neither 'keyed' nor 'relative motion' specified
445	-189	Key already exists and duplicates not allowed
446	-190	Key positioning error
447	-191	Illegal record length
448	-192	Not enough arguments
449	-193	Illegal attribute
450	-194	No debug address
451	-195	Command line too long
452	-196	No starting address
453	-197	Checksum error
454	-198	No source file specified
455	-199	Not a command
456	-200	Illegal block type
457	-201	No files match specifier
458	-202	Phase error
459	-203	Too many arguments
460	-204	Too many active devices
461	-205	Illegal numeric argument
462	-206	Fatal system utility error
463	-207	Illegal argument
464	-208	Improper or malicious input
465	-209	Too many levels of indirect files
466	-210	Syntax error
467	-211	Bracket error
468	-212	Parenthesis error

Table D-4 Business BASIC I/O Error Messages *(continued)*

Record Number in BASIC.ER	Error Code Returned in SYS(7)	Error Message
469	-213	Unmatched <>
470	-214	Illegal nesting of <> and ()
471	-215	Illegal indirect filename
472	-216	Illegal nesting of () and {}
473	-217	Illegal variable
474	-218	Illegal text argument
475	-219	Text argument too long
512	-256	No database record for index entry
513	-257	Node size too big for page
514	-258	Node size will not hold 3 entries
515	-259	Database record is locked
516	-260	Subindex in use encountered during processing
517	-261	File and system versions incompatible
518	-262	Too many subindex links
519	-263	Subindex already present
520	-264	Subindex level limit exceeded
521	-265	Index entry with subindex may not be deleted
522	-266	Physical delete access must be 'keyed'
523	-267	Index entry is locked
524	-268	Write access must be 'keyed'
525	-269	Illegal label format on mag tape
526	-270	Illegal label specification
527	-271	Volume identifiers do not match
528	-272	File identifiers do not match (internal error)
529	-273	File sequence numbers do not match (internal error)
530	-274	File generation numbers do not match (internal error)
531	-275	File expiration date not yet reached
532	-276	Block count on trailer label disagrees with actual count
533	-277	Record formats do not match
534	-278	Incorrect file section number on header label
535	-279	Excessive position levels
536	-280	System load size error
537	-281	Tape file not found
538	-282	Block size less than 8 bytes
539	-283	Record plus overhead is greater than block size
540	-284	Write is not at end-of-file for shared SAM update file
541	-285	Only one user may write to a shared SAM update file

Table D-4 Business BASIC I/O Error Messages *(concluded)*

Record Number in BASIC.ER	Error Code Returned in SYS(7)	Error Message
542	-286	Spooling enabled on illegal device
543	-287	INFOS retrieve key error
544	-288	Delete index positioning error
545	-289	Space management inconsistency
546	-290	Error searching current position table
547	-291	Tape mount cancelled by operator

Table D-5 AOS/VS System Error Messages

Error Code Returned by SYS(31)	System Error Messages
-1	Illegal system command
-2	Channel not open
-3	Channel already open
-4	Shared I/O request not map slot aligned
-5	Insufficient memory available
-6	Illegal starting address
-7	Illegal overlay number
-8	Illegal set time argument
-9	No task control block available
-10	?XMT to address already in use
-11	Error in user task queue table
-12	Task ID error
-13	Data channel map is full
-14	System call parameter address error
-15	Task not found for abort
-16	Insufficient room in buffer
-17	File space exhausted
-18	User stack fault
-19	Directory does not exist
-20	Illegal filename character
-21	File does not exist
-22	File name already exists
-23	Non-directory argument in pathname
-24	End of file
-25	Directory delete error
-26	Write access denied
-27	Read access denied
-28	Append and/or write access denied
-29	No free channels
-30	Release of non-active shared slot
-31	Illegal process priority
-32	Illegal maximum process size
-33	Illegal process type
-34	Console device specification error
-35	Out of swap file room
-36	Device already in system
-37	Illegal device code
-38	Error on shared partition set
-39	Error on remap call
-40	Illegal agent gate call
-41	No PID available for this process
-42	IPC message longer than buffer
-43	Invalid port number
-44	No matching send

Table D-5 AOS/VS System Error Messages *(continued)*

Error Code Returned by SYS(31)	System Error Messages
-45	No outstanding receive
-46	Illegal origin port
-47	Illegal destination port
-48	Invalid shared library reference
-49	Illegal record length specified
-50	Attempt to release console device
-51	Device already in use
-52	Attempt to release unassigned device
-53	Attempt to close unopen channel or device
-54	I/O terminated by CLOSE
-55	Line too long
-56	Parity error
-57	Resident process tried to create son and block
-58	Not a directory
-59	Shared I/O request not to shared area
-60	Too many subordinate processes
-61	File read error
-62	Device timeout
-63	Wrong I/O type for OPEN type
-64	Filename too long
-65	Positioning before beginning of file
-66	Caller not privileged for this action
-67	Simultaneous requests on same channel
-68	Illegal file type
-69	Insufficient room in directory
-70	Illegal OPEN
-71	Attempt to access process not in hierarchy
-72	Attempt to block unblockable process
-73	Invalid system call parameter
-74	Attempt to start multiple agents
-75	Channel in use
-76	Not enough contiguous disk blocks
-77	Stack overflow
-78	Inconsistent bitmap data
-79	Illegal block size for device
-80	Attempt to ?XMT illegal message
-81	Physical unit failure
-82	Physical write lock
-83	Physical unit off-line
-84	Illegal OPEN option for file type
-85	Too many or too few device names
-86	Disk and file system revision numbers don't match
-87	Inconsistent device information block (DIB) data

Table D-5 AOS/VS System Error Messages *(continued)*

Error Code Returned by SYS(31)	System Error Messages
-88	Inconsistent logical disk unit (LDU)
-89	Incomplete logical disk unit (LDU)
-90	Illegal device name type
-91	Illogical process address space definition
-92	LDU in use, cannot release
-93	Too many directories in search list
-94	Cannot get IPC data from father
-95	Illegal library number given
-96	Illegal record format
-97	Too many or too few arguments to PMGR
-98	Illegal ?GTMES parameters
-99	Illegal CLI message
-100	Message receive disabled by CHARACTERISTICS/NRM
-101	Not a console device
-102	Attempt to exceed maximum index level
-103	Illegal channel
-104	No receiver waiting
-105	Short receive request
-106	Transmitter inoperative
-107	Illegal username
-108	Illegal link number
-109	Disk positioning error
-110	Message text longer than specified
-111	Short transmission
-112	Illogical histogram call
-113	Illegal retry value
-114	ASSIGN error - already your device
-115	Mag tape request past logical end of tape
-116	Packet specifies stack too small
-117	Packet would create too many tasks
-118	Spooler open retry count exceeded
-119	Illegal ACL
-120	?STMAP buffer contains invalid or write-protected page
-121	Partner process has not opened IPC file
-122	FPU hardware not installed
-123	Illegal process name
-124	Process name already in use
-125	Disconnect error on modem
-126	Process must block to pass generic files
-127	System not installed
-128	Maximum directory tree depth exceeded
-129	Releasing out-of-use overlay
-130	Resource deadlock
-131	File is open, cannot exclusively open

Table D-5 AOS/VS System Error Messages *(continued)*

Error Code Returned by SYS(31)	System Error Messages
-132	File is exclusively opened, cannot open
-133	Initialization privilege denied
-134	Multiple ?IMSG calls to same DCT
-135	Illegal link
-136	Illegal DUMP format
-137	EXEC not available
-138	EXEC request function unknown
-139	EXEC's process sub-tree only
-140	Request refused by system operator
-141	Cannot dismount, was not mounted
-142	Switch or argument value greater than 65535
-143	Input file does not exist
-144	Output file does not exist
-145	LIST file does not exist
-146	DATA file does not exist
-147	Recursive generic file OPEN failure
-148	No message waiting
-149	User data area (UDA) does not exist
-150	Illegal device type from VSGEN
-151	AOS/VS restart of system call
-152	Fatal user runtime error
-153	User commercial stack fault
-154	User floating point stack fault
-155	User data area (UDA) already exists
-156	Illegal screen_edit request (PMGR)
-157	?SEND destination device held by ^S (CTRL-S)
-158	Data overrun error
-159	Control point directory max size exceeded
-160	System or bootstrap disk not part of master logical disk (LDU)
-161	Universal system, you can't do that
-162	Execute access denied
-163	Can't initialize LDU, run FIXUP over it
-164	File access denied
-165	Directory access denied
-166	Attempt to define greater than one INFOS II process
-167	INFOS II process not running
-168	Attempt to issue MCA request while direct I/O in progress
-169	Attempt to issue MCA direct I/O with request outstanding
-170	Last task was killed
-171	Resource load or release failure
-172	Zero length filename specified
-173	Buffer overflow
-174	Transmission failure (too many NAKS)

Table D-5 AOS/VS System Error Messages *(continued)*

Error Code Returned by SYS(31)	System Error Messages
-175	Transmission failure (timeouts)
-176	Disconnect occurred on sync line
-177	EOT character received
-178	Possible lost data on HASP line
-179	Sync DCU inoperative (I/O aborted)
-180	Conversational reply received
-181	End of polling list reached without a response
-182	Illegal relative terminal number
-183	Reverse interrupt response received
-184	Illegal line number specified
-185	Not enough space for polling list
-186	Contention situation while bidding
-187	Out-of-sequence gen entry during SINIT
-188	Request to a non-synchronous line
-189	Not enough memory for communications buffer
-190	Line already enabled when ?SEBL call issued
-191	Line already disabled when ?SDBL call issued
-192	I/O request on a disabled line
-193	Line in session on initial I/O request
-194	Line not in session on continue I/O request
-195	Buffer byte count larger than system buffer
-196	Bid error (too many NAKS)
-197	Wait-A-Bit received (HASP line only)
-198	User buffer byte pointer invalid
-199	Retry count exceeded
-200	ETX code received
-201	Input status format error
-202	Failure to connect error
-203	Uninterpretable response received
-204	ENQ received
-205	Block check error
-206	Initialization parameter error
-207	Transmitter failure error
-208	Line not multidrop
-209	Not a control station
-210	Polling list not defined
-211	Inconsistent tab format
-212	Cannot delete permanent file
-213	System call abort
-214	Extended context already defined
-215	Unreadable tape or diskette label
-216	Incorrect labeled volume mounted
-217	Incorrect labeled tape fileset
-218	Incorrect file section number

Table D-5 AOS/VS System Error Messages *(continued)*

Error Code Returned by SYS(31)	System Error Messages
-219	Incorrect labeled tape file generation number
-220	Incorrect labeled tape file version number
-221	No operator available
-222	Unknown tape label format
-223	Extended context already initialized
-224	Extended context not initialized
-225	Extended context not defined
-226	Memory release error
-227	Illegal translation parameter
-228	Missing or invalid argument
-229	Not in CLI format
-230	Illegal bias factor
-231	CPU time limit exceeded
-232	Error in setting max CPU limit
-233	File element size not multiple of 4
-234	WACK response received
-235	Process is not a server
-236	Connection does not exist
-237	Connection table full
-238	Directory in use - cannot delete
-239	Attempt to grow a shared file
-240	Illegal directory name specification
-241	Network not available
-242	Host already exist
-243	Illegal host specification
-244	Host does not exist
-245	Cannot rename host entries
-246	Empty mailbox on ?RECNW
-247	Unable to access network in this manner
-248	Attempt to create multiple local host
-249	Not awaiting ?IWKUP
-250	Illegal remote ?PROC parameter(s)
-251	Illegal host name
-252	Not proper for a virtual circuit
-253	HDLC - invalid window size
-254	Invalid frame size
-255	SEND active
-256	Invalid call type
-257	Remote is disconnecting
-258	Local received invalid response
-259	Local received CMDR
-260	Local is in CAN'T SEND condition
-261	Local is disconnecting

Table D-5 AOS/VS System Error Messages *(continued)*

Error Code Returned by SYS(31)	System Error Messages
-262	Local was reset
-263	Buffer overflow
-264	Receive active
-265	Initialization failed
-266	Local received invalid command
-267	Non-HDLC enable attempted
-268	Interrupt wait task already defined
-269	Map slot error
-270	Get buffer error
-271	Sync DCU inoperative
-272	Error opening SLDCU.PR
-273	Error reading SLDCU.PR
-274	Error closing SLDCU.PR
-275	Error getting memory
-276	Unknown error
-277	Connection has been broken
-278	Attempt HDLC call with no DCU 200
-279	Cannot connect to self
-280	No connection
-281	Tape controller does not support this density mode.
-282	Indecipherable tape density.
-283	File/tape density mismatch.
-284	Illegal label level
-285	Illegal label character
-286	Incorrect labelled tape sequence number
-287	Incorrect labelled tape block count
-288	Volid too long or null
-289	Owner ID too long
-290	User labels too long or too many
-291	Record length exceeds block length
-292	AP already busy
-293	AP does not exist
-294	AP WCS is not loaded
-295	Attempt to release a non-AP page
-296	Attempt to release an AP page
-297	Only one file on ANSI level 1 labelled tape
-298	Cannot delete unexpired file on labelled medium
-299	Process console does not exist
-300	Histogram target process termination
-301	Error detected while histogram in progress
-302	Invalid IPC message
-303	Multiple users of file, cannot truncate
-304	Shared file, cannot truncate

Table D-5 AOS/VS System Error Messages *(concluded)*

Error Code Returned by SYS(31)	System Error Messages
-305	File being truncated; cannot open
-306	Framing error
-307	Internal directory inconsistency
-308	Commercial fault
-309	Fixed point fault
-310	Floating point fault
-311	UNKNOWN MESSAGE CODE 00000467
-312	Too many tape retries
-313	No statistics available for this device
-314	No statistics available for this unit
-315	Edit program space exceeded
-316	Edit program too large
-317	Edit program version incompatibility
-318	EDITREAD status error
-319	Illegal EXEC string parameter
-320	Bad internal EXEC IPC
-321	Device code is undefined
-322	No matching ?TLOCK (task was not protected)
-323	Not enough disks of this type selected in VSGEN
-324	Task has an outstanding ?UIDCALL
-325	Task has not been called by ?UIDCALL
-326	File inaccessible, run FIXUP on this LDU
-327	LDU; must have FIXUP run on it
-328	Bad transfer count from controller
-329	Invalid baud rate for this device
-330	Illegal operation for this medium

Table D-6 Business BASIC CLI Error Messages

Record Number in BASIC.ER	Error Code Returned in SYS(7)	Error Message
448	-192	Not enough arguments
449	-193	Illegal attribute
450	-194	No debug address
451	-195	Command line too long
452	-196	No starting address
453	-197	Checksum error
454	-198	No source file specified
455	-199	Not a command
456	-200	Illegal block type
457	-201	No files match specifier
458	-202	Phase error
459	-203	Too many arguments
460	-204	Too many active devices
461	-205	Illegal numeric argument
462	-206	Fatal system utility error
463	-207	Illegal argument
464	-208	Improper or malicious input
465	-209	Too many levels of indirect files
466	-210	Syntax error
467	-211	Bracket error
468	-212	Parenthesis error
469	-213	Unmatched <>
470	-214	Illegal nesting of <> and ()
471	-215	Illegal indirect filename
472	-216	Illegal nesting of () and {}

End of Appendix

Appendix E
Summary Tables of Privileged System Calls

The following tables list the Business BASIC privileged system calls. They also give short descriptions and indicate whether the privileged system calls can be used in AOS or RDOS. Unless a specific section indicates otherwise, these tables use "AOS" to refer to AOS, AOS/WS, AOS/VS, and AOS/VS II. When differences exist, "AOS/VS" and "32-bit only" are used to refer to AOS/VS and AOS/VS II (the 32-bit operating systems), and "AOS" and "16-bit only" are used to refer to AOS and AOS/WS (the 16-bit operating systems). Similarly, "RDOS" refers to RDOS and DG/RDOS unless otherwise noted.

Table E-1 STMB Summary Table

System Call	Description	AOS	RDOS
STMB 0	Retrieves a word address.	Yes	Yes
STMB 1	Copies the contents of memory from an address.	Yes	Yes
STMB 2	Copies words into memory.	Yes	Yes
STMB 3	Copies bytes from memory.	Yes	Yes
STMB 4	Copies the contents of a string into memory.	Yes	Yes
STMB 5	Copies the contents of memory (narrow word or double word).	AOS/VS only	—
STMB 6	Copies 1 or 2 words (up to 32 bits) into memory (AOS/VS). Attaches a detached job to your terminal (RDOS).	AOS/VS only	Yes
STMB 7	Forces a job to BYE off the system.	—	Yes
STMB 8	Resets ON IKEY for a job, clears the "ignore IKEY" flag, and stops the program.	—	Yes
STMB 10	Sets or clears bits in a word.	Yes	Yes

Table E-1 STMB Summary Table *(concluded)*

System Call	Description	AOS	RDOS
STMB 11	Tallies the 1 bits; tallies contiguous 0 bits for a string.	Yes	Yes
STMB 12	Scans the job table for the first available job and executes HELLO for that job. It returns -1 if no available job exists.	Yes	Yes
STMB 13	Places the contents of a string into the input buffer for the specified job.	Yes	Yes
STMB 14	Sets the job to act as if an interrupt occurred.	—	Yes
STMB 15	Sets the multiplexor line characteristics.	—	Yes
STMB 16	Sets or clears the run-only flag in your program.	Yes	Yes
STMB 17	In RDOS, shuts down the multiplexor and executes .RTN. In AOS, identical to STMB 20.	Yes	Yes
STMB 18	Returns the word stored at a specified address in the operating system's address space.	Yes	Yes
STMB 19	Sets the specified multiplexor line to the specified modem status.	—	Yes
STMB 20	Performs an immediate BYE to log off the current job.	Yes	Yes
STMB 21	Not used.	—	—
STMB 22	Returns the byte address of a string.	Yes	—
STMB 23	Returns the word address of a numeric variable.	Yes	—
STMB 24	Performs a system call using information in an accumulator string.	Yes	—
STMB 25	Maps the user channel to the system channel.	Yes	Yes

Table E-2 STMC Summary Table

System Call	Description	AOS	RDOS
STMC 0	Creates a contiguous file with all locations initialized to 0.	Yes	Yes
STMC 1	Creates a subdirectory.	Yes	Yes
STMC 2	Changes the attributes for the file open on a channel.	—	Yes
STMC 3	Changes the link attributes of the file open on a channel.	—	Yes
STMC 4	Returns the 36-byte status table for the file open on a specified channel.	—	Yes
STMC 5	In AOS, creates a control point directory. In RDOS, creates a subpartition.	Yes	Yes
STMC 6	In AOS, creates a UDF file. In RDOS, creates a random file.	Yes	Yes
STMC 7	In AOS, creates a UDF file. In RDOS, creates a sequential file.	Yes	Yes
STMC 8	Deletes a file.	Yes	Yes
STMC 9	Changes the current system directory.	Yes	Yes
STMC 10	Renames a device.	—	Yes
STMC 11	Terminates Business BASIC, passing the error code to the previous level.	Yes	Yes
STMC 12	In AOS, creates a son process and blocks Business BASIC until the son terminates. In RDOS, checkpoints the background and executes the program you specify.	Yes	Yes
STMC 13	In AOS, creates a son process without blocking Business BASIC. In RDOS, executes a program in the foreground partition.	Yes	Yes
STMC 14	In AOS, determines whether a process has sons. In RDOS, determines whether a foreground program is running.	Yes	Yes

Table E-2 STMC Summary Table *(continued)*

System Call	Description	AOS	RDOS
STMC 15	In AOS, this call returns @INPUT. In RDOS, returns the name of the console input device for this ground.	Yes	Yes
STMC 16	In AOS, this call returns @OUTPUT. In RDOS, returns the name of the console output device for this ground.	Yes	Yes
STMC 17	Returns the name of the current Business BASIC system directory.	Yes	Yes
STMC 18	Returns the attributes of the file open on channel.	—	Yes
STMC 19	Returns the current operating system's name.	—	Yes
STMC 20	Initializes a device into the system.	—	Yes
STMC 21	Creates a link entry.	Yes	Yes
STMC 22	In AOS, gives a string a length of zero. In RDOS, returns the name of the current master device.	Yes	Yes
STMC 23	Disables console keyboard interrupts.	—	Yes
STMC 24	Enables console keyboard interrupts.	—	Yes
STMC 25	Renames a file.	Yes	Yes
STMC 26	Closes all files currently open in this ground.	—	Yes
STMC 27	Releases a previously initialized device.	—	Yes
STMC 28	Returns control to the program at the previous push level.	Yes	Yes
STMC 29	Sets the current system date.	—	Yes

Table E-2 STMC Summary Table *(continued)*

System Call	Description	AOS	RDOS
STMC 30	Disables spooling for a device.	—	Yes
STMC 31	Enables spooling for a device.	—	Yes
STMC 32	Kills spooling for a device.	—	Yes
STMC 33	Returns the 18-byte status of the file you specify.	—	Yes
STMC 34	Sets the time of day.	—	Yes
STMC 35	Removes a link entry.	Yes	Yes
STMC 36	Updates the disk-resident copy of the file open on channel.	Yes	Yes
STMC 37	Returns the frequency of the system's real-time clock.	Yes	Yes
STMC 38	Opens a file for direct magnetic tape I/O.	Yes	Yes
STMC 39	Opens a file for appending.	Yes	Yes
STMC 40	Opens a file for exclusive use.	Yes	Yes
STMC 41	In AOS, opens a file for input only. In RDOS, opens a file for read only.	Yes	Yes
STMC 42	In AOS, opens a file for input and output. In RDOS, opens a file for shared access.	Yes	Yes
STMC 43	Terminates Business BASIC and saves a break file.	Yes	Yes
STMC 44	In RDOS, opens a file you specify for exclusive access transparently. In AOS, opens the file for input and output.	Yes	Yes
STMC 45	Transparently creates a contiguous file.	—	Yes

Table E-2 STMC Summary Table *(concluded)*

System Call	Description	AOS	RDOS
STMC 46	Transparently creates a random file.	—	Yes
STMC 47	Transparently creates a sequential file.	—	Yes
STMC 48	Gets the current memory allocation for the current ground.	—	Yes
STMC 50	Writes a line to the file open on a specified channel.	Yes	Yes
STMC 51	Returns the status of the resolution file when you specify a link entry.	—	Yes
STMC 52	Creates a contiguous file without initializing the file to nulls.	Yes	Yes
STMC 53	Closes the file open on a channel.	—	Yes

Table E-3 STMD Summary Table

System Call	Description	AOS	RDOS
STMD 0	Sends a message and can receive a reply, but can't override a no-message flag set by the terminal.	—	Yes
STMD 1	Sends a message and can receive a reply; can also override a no-message flag set by the terminal.	—	Yes

Table E-4 STME Summary Table

System Call	Description	AOS	RDOS
STME 0	Starting from a specified position, returns the next filename matching the template in the directory open on channel.	Yes	—
STME 1	Returns the process's username.	Yes	—
STME 2	Returns the process's PID.	Yes	—
STME 3	Returns a CLI message.	Yes	—
STME 4	Returns the complete pathname of filename.	Yes	—
STME 5	Returns the 23-word (46-byte) status packet for channel in the string variable buffer$.	Yes	—
STME 6	Returns the status of a file.	Yes	—
STME 7	Returns the status of the resolution file when you specify a link entry.	Yes	—
STME 8	Creates a contiguous file.	Yes	—
STME 9	Creates a file.	Yes	—
STME 10	Returns the complete pathname of the file open on channel.	Yes	—
STME 11	Returns the resolution name of a link.	Yes	—
STME 12	Returns the device characteristics of @INPUT.	Yes	—
STME 13	Returns the device characteristics of @OUTPUT.	Yes	—
STME 14	Sets the device characteristics of @INPUT.	Yes	—
STME 15	Sends a message to a specified process.	Yes	—

Table E-4 STME Summary Table *(concluded)*

System Call	Description	AOS	RDOS
STME 16	Sends a message to the process controlling the console.	Yes	—
STME 17	Puts the 256-bit (32-byte) delimiter table for @INPUT into a specified variable.	Yes	—
STME 18	Sets the 256-bit (32-byte) delimiter table for @INPUT.	Yes	—
STME 19	Executes a specified program, passing a string to the new process as the initial IPC.	Yes	—
STME 20	Sends a string via interprocess communications.	Yes	—
STME 21	Receives the contents of a string via interprocess communications.	Yes	—
STME 22	Sends the contents of a string via interprocess communications; then receives an interprocess communications message.	Yes	—
STME 23	An IPC file entry is created with the appropriate name and local port.	Yes	—
STME 24	Finds the owner of a global port.	Yes	—
STME 25	Looks up a port number.	Yes	—
STME 26	Translates a port number.	Yes	—
STME 27	Moves contents of a string to an I/O buffer.	Yes	—

End of Appendix

Appendix F
Terminal Types

The *terminal type* identifies the kind of display device where Business BASIC is being run. The terminal type determines how Business BASIC handles terminal input and output, including the terminal control operations that are possible using the PRINT, STMA, and INPUT statements (refer to the Business BASIC reference manual for commands, statements and functions).

The HELLO program attempts to identify the terminal type when Business BASIC is invoked. If it cannot, it prompts you for the terminal type. The terminal type is displayed in the logon banner as the last character in the account name. You can also use STMA 1,0 to obtain the terminal type.

The CRT models and the corresponding terminal types are given in the table below.

Model	Terminal Type
TELETYPE	0
H.P. 2640/44/45	1
DASHER® HARDCOPY, DECWRITER, TTY	2
6012	3
ADM-1®, ADM-2®	4
HAZELTINE 2000, MOD1	5
DG 605x-compatible terminals	6
ADM-3A®	7
Terminal type 8 (AOS only)	8
Terminal type 9 (AOS only)	9

All hardcopy terminals should use terminal type 2.

Non-Data General Terminals

Data General does not support non-Data General terminals; however, source modules (-CRT-.SR) for the drivers for some non-Data General terminals are provided in the $DOC directory. These modules might not be up-to-date or support current standards for these terminals.

To use one of these modules, move it to BASICGEN or BASICGEN3. To assemble the module in AOS, use the BBUMASM macro provided in the BASICGEN directory of your Business BASIC system. To assemble the module in RDOS, use the RDOS utility MAC. Then edit the LINK (AOS) or RLDR (RDOS) command line (BUILD_system.CLI in AOS or system.CM in RDOS) to add the module's name so that it is included when the system is rebuilt. You must also remove the

corresponding XCRT- module. For example, to include the CRT4 module, you would add CRT4 to the LINK or RLDR line and remove the XCRT4 reference; then execute the LINK or RLDR line to build the system. Under AOS, you should also either regenerate the operating system and describe the characteristics of the non-Data General terminal and the line to which it is attached, or use the AOS CLI CHARACTERISTICS command to set the terminal's characteristics. The second method is performed each time the terminal is used.

The following applies to AOS only:

- Terminal output is buffered up to 512 bytes, thereby decreasing system processing time. Use STMA 8,5 to force output to the screen immediately (without buffering).

- Terminal types 6 and 8 provide the same relative performance for both input and output. Terminal type 6 uses data-sensitive line reads while terminal types 8 and 9 use data-sensitive screen-edit reads.

- Business BASIC attempts to leave the screen environment unchanged. This means that users should use the AOS CLI CHARACTERISTICS command to set desired terminal characteristics before bringing up Business BASIC.

- Applications that capture the cursor positioning keys (^W, ^X, ^Y, ^Z) on input should use terminal type 6 since the ^X and ^Y are Screen Management Primitives that cannot be put in the buffer under terminal type 8. With these applications you must also execute CHAR/OFF/ST/EB0/ON/EB1 before bringing up Business BASIC to cause these keys to move the cursor rather than echoing as ^W, ^X,^Y or ^Z.

  If you want to allow the Escape key to interrupt a program, then execute CHAR/ON/ESC.

  The TAB key (Ctrl-I) is echoed as a tab under either terminal type 6 or 8 unless CHAR/OFF/ST is executed. Ctrl-I may be entered into the buffer.

- Certain control characters have special meanings to AOS and must be preceded by a Ctrl-P on input to be entered into the buffer. These are: Ctrl-C, Ctrl-O, Ctrl-P, Ctrl-Q, Ctrl-S, Ctrl-T, and Ctrl-V.

- Certain control characters with special meaning to AOS cannot be entered into the buffer. These include Ctrl-D, Ctrl-J, Ctrl-K, Ctrl-L, and Ctrl-U.

- If @INPUT/@OUTPUT is not a console, the terminal type is 0.

- All terminal types use New Line as the default primary unpend key and Carriage Return as the secondary unpend key.

Terminal Types 8 and 9

These terminal types serve as interface mechanisms that permit Business BASIC users under AOS with 605x-compatible terminals to use the AOS screen-edit

capability for cursor control. This allows easy redisplay and editing of lines using the AOS screen management primitives (see below), making it possible to bypass the dot (.) editor. In addition, STME 27 is available under terminal types 8 and 9. This lets you edit the input buffer with the screen management primitives. If CHAR/ON/FKT is executed prior to invoking Business BASIC, then, with terminal types 8 and 9, it is possible to terminate an INPUT USING "","",A$,B$ statement with a function key and to capture the two-character function key sequence in B$.

Table F-1 Screen Management Primitives

Key	Function
Ctrl-A	Move to the end of the character string.
Ctrl-B	Move to the end of the previous word.
Ctrl-E	Enter/exit the insert character mode.
Ctrl-F	Move to the beginning of the next word.
Ctrl-H	Move to the beginning of the character string (HOME).
Ctrl-K	Erase everything at and to the right of the cursor (ERASE EOL).
Ctrl-M	Erase everything at and to the right of the cursor and unpend (CR).
Ctrl-X	Move to the right one character.
Ctrl-Y	Move to the left one character.

Terminal types 8 and 9 differ in the default page width. It is 80 for terminal type 8 and 132 for terminal type 9. Also, terminal type 9 has no support for cursor positioning via the PRINT @(x,y) statement, nor does it support the terminal control operations listed in the Terminal Action Codes.

Not all of the functions listed in Table F-2 work for a given terminal. To determine if a function is implemented, type the CRT source module, CRTx.SR (where x is your terminal type). The source modules are located in the $DOC directory. The functions are listed in the module along with the octal representation of the character used to perform the function. A value of -1 indicates that the function is not implemented for that terminal type.

Table F-2 Terminal Action Codes

Key	Function
-19	Reset items -1 to -15 to their default values for your terminal. If you change your terminal type with STMA 2,0 and want the new terminal type's default characteristics, follow the STMA 2,0 with PRINT @(-19).
-20	Move cursor to the first character position on the top line (home position) of the display screen.
-21	Move cursor right one column.
-22	Move cursor down one column.
-23	Move cursor left one column.
-24	Move cursor up one column.
-25	Sound the bell or audible alarm.
-26	Tab to the next tab stop.
-27	Return cursor to first character position of the current line.
-28	Move cursor to the beginning of the next line.
-29	Back tab.
-30	Clear the screen (or generate a form feed in a file).
-31	Clear unprotected positions of the screen (high intensity).
-32	Clear the screen to the end of the line.
-33	Clear to the end of the screen.
-34	Lock keyboard.
-35	Unlock keyboard.
-36	Insert a line.
-37	Delete a line.
-38	Start displaying text in low intensity.
-39	Start high-intensity field.

Table F-2 Terminal Action Codes *(concluded)*

Key	Function
-40	Start displaying blinking text.
-41	End displaying blinking text.
-42	Set page mode.
-43	Turn off page mode.
-44	Set program mode.
-45	Clear program mode.
-46	Set block mode.
-47	Clear block mode.
-48	Set flag 1.
-49	Clear flag 1.
-50	Send line (unprotected fields).
-51	Send line (all fields).
-52	Begin underlining text.
-53	End underlining text.
-54	Start field display enhancement.
-55	Start literal display enhancement.
-56	Restore default display enhancement.

Terminal types 8 and 9 must be requested when Business BASIC is generated, and they must be enabled by invoking Business BASIC with a /E global switch.

There are certain restrictions which apply to terminal types 8 and 9:

- The delete key (DEL) may not be redefined.
- The AOS line cancel key (Ctrl-U) may not be redefined.
- Even if control characters are allowed on input, the Screen Management Primitives in the Terminal Action Codes table above cannot be put into the buffer.

- If you are allowing control characters on input, and you input a Ctrl-D, the input is terminated and the value of SYS(10) is not changed.

- If you are allowing only uppercase characters (each lowercase character input is translated into uppercase), and you define your unpend character to be an uppercase alphabetic character (for example, A), inputting the string 1a2bA echoes to the terminal as 12B and unpending takes place. The buffer, however, contains the string 1A2B. This is due to an AOS restriction.

- If you are allowing control characters on input, the way they are echoed on the screen is governed by the settings of the AOS characteristics EB0 and EB1. If EB0 is on and EB1 is off, the control characters are echoed with carets (for example, Ctrl-W as ^W).

- If a Ctrl-E operation is in progress and a character is input that should be ignored (for example, a control character), the Ctrl-E is terminated and the cursor is placed at the end of the buffer.

- If the input characters span more than one line on the screen, the cursor position at the end of a read or input statement is left wherever the cursor was when the unpend key was pressed. If the input characters do not span more than one line, the cursor is left at the greater of either the ending column position or the starting column number plus the number of input characters.

- Programs must not use STMB 24 to make a ?SDLM call to change the delimiter table since terminal type 8 sets the delimiter table for its own purposes.

End of Appendix

Index

Note: Boldfaced page numbers (e.g., **1-5**) indicate definitions of terms or other key information.

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Related Documents

Business BASIC Reference Manual for Commands, Statements, and Functions, 093–000351

An alphabetical directory of Business BASIC commands, statements, and functions. It is intended to be used as a reference manual for programmers.

Business BASIC Reference Manual for Subroutines, Utilities, and BASIC CLI, 093–000389

An alphabetical directory of Business BASIC subroutines, utilities, and BASIC CLI commands. It is intended to be used as a reference manual for programmers.

Business BASIC Summary, 069-000263

A booklet summarizing Business BASIC's commands, statements, functions, subroutines, and BASIC CLI commands.

Programming with Business BASIC, 093-000480

A manual for experienced programmers who have not used Business BASIC. The purpose is to acquaint these programmers with Business BASIC operations and programming procedures. This manual provides an overview of the Business BASIC commands, functions, subroutines, and utilities available to programmers.

DASHER® D2 File Maintenance and Screen Maintenance Template, 093-000212
DASHER® D200 File Maintenance and Screen Maintenance Template, 093-000265
DASHER® D210/211 D410/460 CFM and CSM Template, 093-000409
DASHER® D211/211 D410/460 SM and FM Template, 093-000410

AOS INFOS® II System User's Manual, 093-000152

A guide to using the AOS INFOS® II file management system.

AOS/VS INFOS® II System User's Manual, 093-000299

A guide to using the AOS/VS INFOS® II file management system.

AOS/VS System Concepts, 093-000335

System Call Dictionary (AOS/VS and AOS/DVS), 093-000241

AOS/VS Macroassembler (MASM) Reference Manual, 093-000242

Eclipse 32-Bit Systems Principles of Operation Programmer's Reference, 014-000704

How to Load and Generate RDOS, 069-400013

How to Generate and Run DG/RDOS, 093-000470

TP _____

TIPS ORDER FORM

Mail To: Data General Corporation
Attn: Educational Services/TIPS G155
4400 Computer Drive
Westboro, MA 01581 - 9973

BILL TO:	SHIP TO: (No P.O. Boxes - Complete Only If Different Address)
COMPANY NAME _____	COMPANY NAME _____
ATTN: _____	ATTN: _____
ADDRESS _____	ADDRESS (NO PO BOXES) _____
CITY _____	CITY _____
STATE _____ ZIP _____	STATE _____ ZIP _____

Priority Code _____ (See label on back of catalog)

Authorized Signature of Buyer (Agrees to terms & conditions on reverse side)	Title	Date	Phone (Area Code)	Ext.

ORDER #	QTY	DESCRIPTION	UNIT PRICE	TOTAL PRICE

A SHIPPING & HANDLING

☐ UPS	ADD
1-4 Items	$ 5.00
5-10 Items	$ 8.00
11-40 Items	$ 10.00
41-200 Items	$ 30.00
200+ Items	$100.00

Check for faster delivery

Additional charge to be determined at time of shipment and added to your bill.

☐ UPS Blue Label (2 day shipping)
☐ Red Label (overnight shipping)

B VOLUME DISCOUNTS

Order Amount	Save
$0 - $149.99	0%
$150 - $499.99	10%
Over $500.00	20%

Tax Exempt # or Sales Tax (if applicable) _____

ORDER TOTAL	
Less Discount See B	-
SUB TOTAL	
Your local* sales tax	+
Shipping and handling - See A	+
TOTAL - See C	

C PAYMENT METHOD

☐ Purchase Order Attached ($50 minimum)
P.O. number is _____. (Include hardcopy P.O.)
☐ Check or Money Order Enclosed
☐ Visa ☐ MasterCard ($20 minimum on credit cards)

Account Number ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ Expiration Date ☐☐☐☐

_____
Authorized Signature
(Credit card orders without signature and expiration date cannot be processed.)

THANK YOU FOR YOUR ORDER

PRICES SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.
PLEASE ALLOW 2 WEEKS FOR DELIVERY.
NO REFUNDS NO RETURNS.

* Data General is required by law to collect applicable sales or use tax on all purchases shipped to states where DG maintains a place of business, which covers all 50 states. Please include your local taxes when determining the total value of your order. If you are uncertain about the correct tax amount, please call 617-870-1600.

Form 702
Rev. 8/87

DATA GENERAL CORPORATION
TECHNICAL INFORMATION AND PUBLICATIONS SERVICE
TERMS AND CONDITIONS

Data General Corporation ("DGC") provides its Technical Information and Publications Service (TIPS) solely in accordance with the following terms and conditions and more specifically to the Customer signing the Educational Services TIPS Order Form. These terms and conditions apply to all orders, telephone, telex, or mail. By accepting these products the Customer accepts and agrees to be bound by these terms and conditions.

1. CUSTOMER CERTIFICATION
Customer hereby certifies that it is the owner or lessee of the DGC equipment and/or licensee/sub-licensee of the software which is the subject matter of the publication(s) ordered hereunder.

2. TAXES
Customer shall be responsible for all taxes, including taxes paid or payable by DGC for products or services supplied under this Agreement, exclusive of taxes based on DGC's net income, unless Customer provides written proof of exemption.

3. DATA AND PROPRIETARY RIGHTS
Portions of the publications and materials supplied under this Agreement are proprietary and will be so marked. Customer shall abide by such markings. DGC retains for itself exclusively all proprietary rights (including manufacturing rights) in and to all designs, engineering details and other data pertaining to the products described in such publication. Licensed software materials are provided pursuant to the terms and conditions of the Program License Agreement (PLA) between the Customer and DGC and such PLA is made a part of and incorporated into this Agreement by reference. A copyright notice on any data by itself does not constitute or evidence a publication or public disclosure.

4. LIMITED MEDIA WARRANTY
DGC warrants the CLI Macros media, provided by DGC to the Customer under this Agreement, against physical defects for a period of ninety (90) days from the date of shipment by DGC. DGC will replace defective media at no charge to you, provided it is returned postage prepaid to DGC within the ninety (90) day warranty period. This shall be your exclusive remedy and DGC's sole obligation and liability for defective media. This limited media warranty does not apply if the media has been damaged by accident, abuse or misuse.

5. DISCLAIMER OF WARRANTY
EXCEPT FOR THE LIMITED MEDIA WARRANTY NOTED ABOVE, DGC MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE ON ANY OF THE PUBLICATIONS, CLI MACROS OR MATERIALS SUPPLIED HEREUNDER.

6. LIMITATION OF LIABILITY
A. CUSTOMER AGREES THAT DGC'S LIABILITY, IF ANY, FOR DAMAGES, INCLUDING BUT NOT LIMITED TO LIABILITY ARISING OUT OF CONTRACT, NEGLIGENCE, STRICT LIABILITY IN TORT OR WARRANTY SHALL NOT EXCEED THE CHARGES PAID BY CUSTOMER FOR THE PARTICULAR PUBLICATION OR CLI MACRO INVOLVED. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO CLAIMS FOR PERSONAL INJURY CAUSED SOLELY BY DGC'S NEGLIGENCE. OTHER THAN THE CHARGES REFERENCED HEREIN, IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO LOST PROFITS AND DAMAGES RESULTING FROM LOSS OF USE, OR LOST DATA, OR DELIVERY DELAYS, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY THEREOF; OR FOR ANY CLAIM BY ANY THIRD PARTY.

B. ANY ACTION AGAINST DGC MUST BE COMMENCED WITHIN ONE (1) YEAR AFTER THE CAUSE OF ACTION ACCRUES.

7. GENERAL
A valid contract binding upon DGC will come into being only at the time of DGC's acceptance of the referenced Educational Services Order Form. Such contract is governed by the laws of the Commonwealth of Massachusetts, excluding its conflict of law rules. Such contract is not assignable. These terms and conditions constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes all prior oral or written communications, agreements and understandings. These terms and conditions shall prevail notwithstanding any different, conflicting or additional terms and conditions which may appear on any order submitted by Customer. DGC hereby rejects all such different, conflicting, or additional terms.

8. IMPORTANT NOTICE REGARDING AOS/VS INTERNALS SERIES (ORDER #1865 & #1875)
Customer understands that information and material presented in the AOS/VS Internals Series documents may be specific to a particular revision of the product. Consequently user programs or systems based on this information and material may be revision-locked and may not function properly with prior or future revisions of the product. Therefore, Data General makes no representations as to the utility of this information and material beyond the current revision level which is the subject of the manual. Any use thereof by you or your company is at your own risk. Data General disclaims any liability arising from any such use and I and my company (Customer) hold Data General completely harmless therefrom.

TIPS ORDERING PROCEDURES

TO ORDER

1. An order can be placed with the TIPS group in two ways:
   a) MAIL ORDER – Use the order form on the opposite page and fill in all requested information. Be sure to include shipping charges and local sales tax. If applicable, write in your tax exempt number in the space provided on the order form.

      Send your order form with payment to:

      Data General Corporation
      ATTN: Educational Services/TIPS G155
      4400 Computer Drive
      Westboro, MA 01581-9973

   b) TELEPHONE – Call TIPS at (508) 870-1600 for all orders that will be charged by credit card or paid for by purchase orders over $50.00. Operators are available from 8:30 AM to 5:00 PM EST.

METHOD OF PAYMENT

2. As a customer, you have several payment options:
   a) Purchase Order – Minimum of $50. If ordering by mail, a hard copy of the purchase order must accompany order.
   b) Check or Money Order – Make payable to Data General Corporation.
   c) Credit Card – A minimum order of $20 is required for Mastercard or Visa orders.

SHIPPING

3. To determine the charge for UPS shipping and handling, check the total quantity of units in your order and refer to the following chart:

Total Quantity	Shipping & Handling Charge
1-4 Units	$5.00
5-10 Units	$8.00
11-40 Units	$10.00
41-200 Units	$30.00
Over 200 Units	$100.00

   If overnight or second day shipment is desired, this information should be indicated on the order form. A separate charge will be determined at time of shipment and added to your bill.

VOLUME DISCOUNTS

4. The TIPS discount schedule is based upon the total value of the order.

Order Amount	Discount
$1-$149.99	0%
$150-$499.99	10%
Over $500	20%

TERMS AND CONDITIONS

5. Read the TIPS terms and conditions on the reverse side of the order form carefully. These must be adhered to at all times.

DELIVERY

6. Allow at least two weeks for delivery.

RETURNS

7. Items ordered through the TIPS catalog may not be returned for credit.
8. Order discrepancies must be reported within 15 days of shipment date. Contact your TIPS Administrator at (508) 870-1600 to notify the TIPS department of any problems.

INTERNATIONAL ORDERS

9. Customers outside of the United States must obtain documentation from their local Data General Subsidiary or Representative. Any TIPS orders received by Data General U.S. Headquarters will be forwarded to the appropriate DG Subsidiary or Representative for processing.

134-755-02

moisten & seal

CUSTOMER DOCUMENTATION COMMENT FORM

Your Name _____ Your Title _____

Company _____ Phone _____

Street _____

City _____ State _____ Zip _____

We wrote this book for you, and we made certain assumptions about who you are and how you would use it. Your comments will help us correct our assumptions and improve the manual. Please take a few minutes to respond. Thank you.

Manual Title _____ Manual No. _____

Who are you? ☐EDP/MIS Manager ☐Analyst/Programmer ☐Other _____
☐Senior Systems Analyst ☐Operator _____
☐Engineer ☐End User

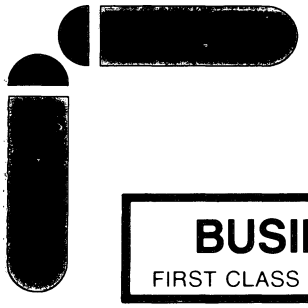
How do you use this manual? *(List in order: 1 = Primary Use)*

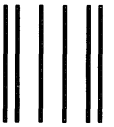
___ Introduction to the product ___ Tutorial Text ___ Other
___ Reference ___ Operating Guide _____

		Yes	No
About the manual:	Is it easy to read?	☐	☐
	Is it easy to understand?	☐	☐
	Are the topics logically organized?	☐	☐
	Is the technical information accurate?	☐	☐
	Can you easily find what you want?	☐	☐
	Does it tell you everything you need to know?	☐	☐
	Do the illustrations help you?	☐	☐

If you wish to order manuals, use the enclosed TIPS Order Form (USA only) or contact your sales representative or dealer.

Comments:





NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 26 WESTBORO, MA 01581

POSTAGE WILL BE PAID BY ADDRESSEE

DataGeneral

Customer Documentation
MS E-111
4400 Computer Drive
P.O. Box 4400
Westboro, MA 01581-9890

134-755-02

Cut here and insert in binder spine pocket

Data General

Data General Corporation, Westboro, Massachusetts 01580

093-000388-01