Software Release Notice

Business BASIC for AViiON Systems

Revision 1.02

Model Number L005ASU

August 1990

085-600140-02

Restricted Rights Legend

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at [FAR]52.227-7013 (May 1987).

DATA GENERAL CORPORATION
4400 Computer Drive
Westboro, MA. 01580

Table of Contents

1 Introduction

The purpose of this release notice is to provide you with information about revision 1.02 of Business BASIC for AViiON Systems which is not available in the Business BASIC for AViiON Systems documentation.

This product consists of the following parts:

Part Description	Part Number
1. Business BASIC for AViiON release notice	085-600140-02
2. Business BASIC for AViiON release media	See section 7 of this notice

This printed release notice always accompanies the software. You can print additional copies of this notice after you have installed the product. Its filename is 085600140_02.

2 Product Description

Data General's Business BASIC is a powerful interactive programming language designed for use in producing business applications. Business BASIC lets multiple users program, test, debug and execute programs independently. It provides many of the standard BASIC commands, statements, and functions. It also contains specialized statements and functions that handle file access, format control, and system functions.

Business BASIC operates under Data General's AOS/VS, DG/UX, and DG/RDOS operating systems on Data General machines ranging from the DESKTOP GENERATION Model 10 to the MV/40000 to the AViiON line. It also operates under INTERACTIVE Systems Corporation's 386/ix on the DASHER/386. INTERACTIVE's 386/ix is an enhanced, high performance UNIX operating system.

Business BASIC supports numerous file structures, including ISAM files, logical files, subfiles within a master file, and linked-available-record files. Under the AOS/VS operating system, it includes an interface to Data General's INFOS II system, which allows Business BASIC statements to manipulate INFOS II files. An interface is also furnished for user-written assembly-language routines under the AOS/VS and DG/RDOS operating systems. Under DG/UX and the 386/ix operating system, an interface is provided for user-written C language routines.

Among the new features of this release are:

* Performance has been substantially improved in a variety of areas, resulting in improved overall throughput. These performance improvements mandated a save file change.

* IPC support has been added with the addition of STMEs 20-26.

* Shared I/O support has been modified to reduce substantially the system resources required to support it.

* Several new options have been added to the bbux_mgr script to allow users to flush shared pages, monitor paging activity, and produce a debug dump file.

* A new numeric function, ASX, is introduced in this revision. This function returns the ASCII value of a string, and automatically handles the sign when extracting a negative number.

* The bb_port utility has been enhanced to enable users to produce listing files from save files.

* TRACE has been enhanced with the addition of an option to send its output to a file.

* A new STMA returns the entire login name up to 32 characters.

* A new STMU provides users with a way to obtain the contents of an environment variable.

* Business BASIC Management now lets you choose whether to automatically start the daemons at system boot time.

* Several older DG terminals are now supported in ANSI mode. These include the D210, D211, D214, D215, D410, D411, D460 and D461.

3 Environment

Prerequisites:

DG/UX Rev. 4.30 or later (one of these two models: Q001A or P001A).

Terminals supported include the AViiON console, the D216, D216E, D412, D462, D216+, D412+, and the D462+. Several of the older Data General terminals are supported in ANSI mode. These include the D210, D211, D214, D215, D410, D411, D460, and the D461.

4 Enhancements and Changes

4.1 Enhancements

Business Basic 1.02 provides several enhancements:

1) A number of changes have been made internally to improve performance in various areas. These changes have speeded up FOR/NEXT loops, block IF statements, math routines, PRINT FILE, and certain PACK formats, to name a few. These improvements forced a change in the save file format. The bb_port utility furnished with Business BASIC for AViiON Systems may be used to help in the conversion process.

2) Support has been added for STME's 20-26 to allow Business BASIC programs to communicate. The syntax of the statements is unchanged so that code which operates under AOS/VS will run correctly under DG/UX , except for one possible situation.

   Several of the STME's (20, 21, 22 and 24) require a global port number either as an argument or as part of the control string. This is generally obtained via an STME 25 or STME 26. If the global port has been gotten through one of these STME's, there is no problem. However, any code which composes the global port value by putting a PID and a local port into a string with PACK or CHR$ will require modification to obtain the global port value from STME 25 or STME 26. Under AViiON BB the global port uses an internal packet index value instead of a PID.

3) Several enhancements have been made to the bbux_mgr script (also known as the BB daemon_mgmt menu reached through sysadm bbasic_mgmt). Three new options have been added, and some modifications have been made to existing options.

   The three new options are:

   * Flush shared pages (option 9)

   * Monitor paging activity (option 10)

   * Create debug dump file (option 11)

   The "flush shared pages" option causes obit to flush all shared pages of open files to disk.

The "monitor paging activity" option is useful for optimizing shared I/O performance. When you choose this option, you are asked to enter the obit directory, the number of iterations (default is 10), and the number of seconds between iterations (default is 5). "Iterations" means the number of samplings to be taken. The output is displayed in three columns: iteration, reads and writes. The iteration column simply identifies each iteration. The reads column shows the number of times since the last iteration that a page had to be read into obit's shared memory segment in order to be accessed by a BB process. The writes column shows the number of times since the last iteration that a page in obit's shared memory segment had to be written out so that another page could be read in. This paging activity is directly influenced by the value you choose for "maximum number of 2048-byte shared file pages" in your obit limits file. Increasing this value should decrease the number of reads and writes. The optimal value will be the smallest value such that a larger value will not decrease the reads and writes. You can try steadily increasing values for the number of pages and monitor the paging activity for those values. When the reads and writes no longer decrease, you are in the optimal range. If you wish, you can then use further experimentation to fine-tune your pages value by lowering it slightly until the reads and writes again increase to detect the point at which you entered the optimal range.

The "create debug dump file" option creates a diagnostic file describing the current state of the rlsx and bbux processes connected to obit. This file would be very useful for Data General support personnel if some problem involving these processes should occur.

Modifications to other options include options 1, 4 and 8. Options 1 and 4 bring up obit and rlsx, respectively. These options now have a default filename for messages from obit and rlsx. The default filenames are obit.msg and rlsx.msg, respectively. If you want messages to go to the console, type /dev/console in response to the prompts for messages files.

Option 8 (calculate kernel parameters) now includes a CHANGE column. If the values in your obit limits file are such that the default DG/UX values for affected tunable parameters will not suffice, a value will appear in the CHANGE column; this is the recommended minimum value you should use for the parameter. If there are other processes which use the system resources affected by these parameters, the parameter settings may have to be higher. If no value appears in the CHANGE column for the parameter,

you do not need to modify that parameter for obit's needs.

Keep in mind that if other applications will be running, they may have requirements which affect the same tunable parameters obit affects. This could cause problems bringing up Business BASIC even if you reconfigured the kernel with the values in the CHANGE column. An example is X, which uses a semaphore. If you set the SEMMNI tunable parameter as recommended in the CHANGE column and run X, you may not be able to bring up Business BASIC; you would have to set SEMMNI to the value recommended in CHANGE plus one for X. Use the ipcs command before bringing up obit and rlsx to see if any other processes are using shared memory segments, message queues, or semaphores.

4) The bb_port utility has been enhanced by adding an option to produce listing files from Business BASIC save files. This change did not affect any existing functionality in bb_port. To use the new option, a minimal amount of additional setup may be necessary:

* If you are using bb_port to convert rev. 1.00 save files, the 'ptd' daemon must be on the BBPATH environment variable.

* On the 386/ix platform, filenames should be 11 characters or less in order for the .LS extension to be added to the source listing names. If this is not the case, bb_port issues an error message and stops.

The only other requirement is that you have correctly set up your system to run Business BASIC as described in this release notice and the "Using Business BASIC on UNIX" manual.

The save-to-list conversion is selected with the -s switch. The syntax of the command line is thus:

```
bb_port [-s -d<dir>] [-l] [-p<path>] [filename...]
```

-s This option will execute the save file conversion part of bb_port. When this is not set, bb_port will convert source listings to save files as in bb_port revision 1.01.

The other options are unchanged from revision 1.01, and are described in the "Using Business BASIC on UNIX" manual. However, when -s is selected, the -d option is required:

-d<dir> This option must be specified when doing save-to-list conversion to specify the directory where the list files will be created. The name of the directory should follow -d with no spaces in between.

NOTE: bb_port will attempt to process all files in the current directory. Any file which is not an acceptable save file will return a 'not a save file' error in port.er. It is also possible for bb_port to return an error if the filename argument is a template which expands to more than about 200 filenames. In this case, the error "awk: argument too long" is returned. Nothing is harmed by this error; you simply must issue a command line with a more restrictive template which selects fewer filenames, and use bb_port more than once.

See the "Using Business BASIC on UNIX" manual for complete information on the other options of bb_port and general guidelines for running it.

5) A new numeric function, ASX, is introduced in this revision. This function is similar to the ASC function in that it returns the ASCII value of a string. The difference is that it automatically corrects the sign when extracting negative numbers. See the documentation update file for "Using Business BASIC on DG/UX and 386/ix Systems" for more detailed information. It is in the DOC directory in the file named u093000685.01.

APERM.PS has been updated to include ASX as a reserved word.

6) INDEXVRFY has been enhanced to verify the backward links in a 2048-byte block index.

7) Two new example scripts are available in the AViiON BB BIN directory. One script, ro_profile.ex, is an example of a run-only profile. It does all the initialization required by Business BASIC, and then does an exec of the interpreter; this runs Business BASIC without a shell (equivalent to a CHAIN in the AOS/VS CLI.) The other script, make_chng.ex, is an example of a script which may be used to change occurrences of a text string in a number of files. It could be used to change the name of a variable in a group of LIST files, for example.

8) Several older DG terminals are now supported in ANSI 8-bit mode. These include the D210, D211, D214, D215, D410, D411, D460 and D461. If these terminals are run in 7-bit

mode, the cursor keys and function keys may not work correctly.

Modified terminfo files for use with these terminals are included in this release. Their names are d210_bbux, d211_bbux, etc.; they are in the AViiON BB TERM/d directory.

9) The TRACE statement/command has been enhanced so that its output may be directed to a file. The syntax is:

```
TRACE ON[,"filename"]
```

where the filename argument is optional. Output from TRACE will then go to the file rather than the screen. This can significantly aid in debugging time. Note that executing TRACE ON with a filename while tracing is already on and being directed to a file will cause the first file to be closed. For example,

```
500 TRACE ON,"TRACEOUT1"
:
:
:
900 TRACE ON,"TRACEOUT2"
```

TRACEOUT1 will be closed when line 900 is executed before tracing output begins going to TRACEOUT2.

10) A new option has been added to STMA 9 to return the entire login name up to 32 bytes. The syntax is:

```
STMA 9,6,X$
```

The terminal type is not included in this string.

11) A new STMU allows users to obtain the contents of an environment variable. Its syntax is:

```
STMU 5,er,envvar$,contents$
```

where er is the error variable, envvar$ is the name of the environment variable, and contents$ receives the environment variable's contents. A -1 is returned in er if no error occurred. Contents$ should be dimensioned large enough to receive the contents of the environment variable requested, or an error 34, "Function argument" error is returned.

12) The Business BASIC Management menu in sysadm has been enhanced with the addition of an option which allows you to

select whether the Business BASIC daemons will start automatically at system boot time. The new option is number 4 in the menu, and is called "init options".

When the user selects this option, something like the following is displayed:

```
Would you like to:
 1) Have the BB daemons start automatically?
 2) No longer start the BB daemons automatically?
The current setting is:
    Do not automatically start the daemons at init level 2
```

The information displayed regarding the current setting will either state that the daemons are not automatically being started, or that they are.

If you choose to have the daemons start automatically, you will then be asked a series of questions. These questions are discussed in section 8.2 of this release notice.

4.2 Changes

1) This revision of Business BASIC features a new implementation of shared I/O support which no longer requires a large amount of swap space for obit's use. Your swap space should be the size required by DG/UX to support the number of users you have with the amount of memory you have. See the discussion on this subject in Chapter 2 of "Installing and Managing the DG/UX System" (093-701052).

2) The save file format of AViiON BB 1.02 is different from that used in earlier revisions of Business BASIC for AViiON Systems. It also is incompatible with that used by AOS/VS and DG/RDOS Business BASIC. Users with rev. 1.00 or rev. 1.01 save files will need to convert them to rev. 1.02 save files via the ENTER/SAVE mechanism. Users coming from AOS/VS or DG/RDOS will need to import source listings of their programs and likewise use the ENTER/SAVE mechanism to produce Business BASIC save files. The bb_port utility furnished with Rev. 1.02 has been enhanced to convert earlier revisions of Business BASIC for AViiON Systems save files to list files. See the Enhancements sections for more information.

   Note: a module named li.a is furnished in the CONVERT directory which corrects a problem in LIST in revision 1.01 of Business BASIC. This problem would cause a core dump if an attempt was made to LIST to a filename, and the program being listed had GOTOs or GOSUBs to non-existant line numbers. If you are converting from rev. 1.01, you should first install this module in 1.01 by putting it in the 1.01 BASICGEN/lb directory, and then building a new 1.01 Business BASIC interpreter by use of the build_bbasic script. This new interpreter may then be used to produce the needed LIST files.

   Programs developed under Revision 1.02 of Business BASIC for AViiON Systems which are to be run under the AOS/VS or DG/RDOS operating systems must be exported as source listings.

3) If a BB process which was doing shared I/O was aborted or killed, it was possible to lose data. This has been corrected. [No STR]

4) A memory leak problem in previous revisions caused memory to be allocated but not freed when a lock was made on a file which had previously been locked. This has been corrected. [No STR]

5) If an attempt was made to LIST"@LPT", a file named @LPT was created. This has been corrected so that the DEVICE_MAP file is used so that output may be directed to the printer. [STR NASC-6190]

6) If an attempt is made to connect a 1.02 BB process or 1.02 rlsx to an pre-1.02 obit, an "incompatible revision of obit" message will be returned.

7) A problem could occur when adding keys to an 2048-byte block index in random order which resulted in the backward links not being properly set up. This could only be seen when using KPREV, and did not affect the index in any other way. The problem has been corrected. [No STR] If your programs use KPREV, you should rebuild your 2048-byte block index files.

8) STMA 9,0 has been modified to work exactly as it does in the AOS/VS version of Business BASIC. It will return a six-byte string, with the first 5 bytes being made up of the first 5 bytes of the login name and the 6th byte being the terminal type. If the login name is less than 5 bytes long, the name will be padded with spaces up to 5 bytes in the returned string. [STR UKSC-40075]

9) GPOS would return an error when used on a channel on which the printer queue file was opened. This has been modified so that a zero will be returned on a queue file, PROVIDED THAT the queue filename is in the DEVICE_MAP file. [STR NASC-5245]

10) PRINT USING "OD",NUM was returning the wrong error message. The error returned was "Illegal channel number" when it should have been "Error 35 - Illegal format string". This has been corrected. [STR NASC-6186]

11) Four additional questions are asked when you define an obit limits file through sysadm bbasic_mgmt. These questions are explained in the "Installation Instructions" section of this release notice.

12) A locking problem which could result in a deadly embrace has been corrected. [No STR]

13) The bbasic_build script now links in the standard DG/UX libcurses.a library. Business BASIC no longer includes a libcurses library.

14) STMA 8,6 was not correctly resetting the DO stack. This has been corrected [No STR].

Note that the documentation for STMA 8,6 was incorrect. It stated that it would pass control to the statement following the outermost DO loop. All it does is reset the DO stack.

15) STMA 8,0 now also resets the DO stack in addition to the FOR...NEXT and GOSUB...RETURN stacks.

16) READ FILE into a second subscripted string was not working correctly. If this code was executed, A$(101,200) would not get data:

```
READ FILE(0),A$(1,100),A$(101,200)
```

This has been corrected. [STR RTP0-2243]

17) STMA 12 would cause a hang if the month variable was zero. This has been corrected. [STR RTP0-2249]

18) Making a call to a UCALL would cause a hang, whether or not the UCALL existed. This has been corrected. [NASC-5654]

19) !DIR would display ::BBTEMP as the current directory if you were in /BBTEMP and issued these commands:

```
* !DIR ..
* !DIR BBTEMP
* !DIR
```

This has been corrected. [STR SQA0-13521]

20) PRINT "<138>" or other characters with the 8th bit set would cause garbaging of the screen. This has been corrected. The 8th bit is now stripped before the character is printed. [STR UKSC-37818]

21) Support of embedded control characters was enhanced to include <11>, to erase to the end of the line. [STR UKSC-38392]

22) LFU LRENAME and LFU LDELETE would return I/O error 10 - file does not exist - for a file which did exist in the current directory. This has been corrected. [STR UKSC-40046]

23) A problem in LIST would cause a core dump if an attempt was made to LIST to a filename, and the program being listed had GOTOs or GOSUBs to non-existant line numbers with the GOTOs or GOSUBs located inside either FOR/NEXT, DO/WHILE or block IF loops. This has been corrected [No STR].

24) The default size for the rlsx lock table has been changed to 8192.

25) STMC 50,1,S$ has been corrected to enforce maximum string length of 133 to match the behavior of AOS/VS Business BASIC. The documentation was in error and is corrected in the documentation update file accompanying this release.

26) ENTERing a program that contains a syntax error would not redisplay the line so corrections could be made. Pressing CTRL-A would display the last line of the program that was successfully ENTERed, not the last line that contained an error. This has been corrected. [No STR]

27) It was possible during automatic booting of the daemons for the script to try to bring up rlsx before obit was completely up. This has been corrected. [STR ITAL-82]

28) A problem with the file pointer being in the wrong position after a PRINT FILE has been corrected. This would cause GPOS on a file opened in mode 5 to return an incorrect value after the PRINT FILE. [STR ITAL-79]

29) APERM.PS has been modified to add ASX and to remove CREATE.

30) It was possible to add more keys to an index than the index was supposed to hold. This has been corrected. [No STR]

4.3 New BB Features and Extensions Introduced in Rev. 1.00

Revision 1.00 of AViiON BB introduced a number of new features and expanded numerous limits in Business BASIC. These were described in the rev. 1.00 release notice. If rev. 1.02 is your first use of UNIX Business BASIC, you may not be aware of the new features and extensions of the language. They are summarized here.

* More user program and data space is now available than in AOS/VS or DG/RDOS versions of Business BASIC. The program space and data space may each be up to 512 Kb, with a total user program and data space of 512 Kb. This allows small programs with large amounts of data, as well as very large programs with small amounts of data.

* Line numbers up to 99999 are now supported.

* The number of variables has been increased from 348 to 8192.

* The length of variable names has been increased from 6 to 32 characters.

* The underscore is now a legal character for a variable name.

* The common area has been increased in size from 512 to 2048 bytes.

* GOSUB nesting has been increased from 8 to 32 levels.

* FOR/NEXT nesting has been increased from 8 to 32 levels.

* A DO WHILE family of statements has been added to the language, making it possible to write more structured code.

* A generic method of dealing with function keys has been introduced using SYS(50) and SYS(51).

* String arrays of up to eight dimensions are supported.

* Numeric arrays have been extended to allow eight dimensions.

* An new XOR logical function has been added to Business BASIC to perform an exclusive OR between two expressions.

* A new COMP logical function has been added to Business BASIC to return the one's complement of an expression.

* SWAPs can now be done in memory instead of on disk, speeding execution. This is selected during Business BASIC sysgen.

* New STMA 22 statements give the Business BASIC programmer the ability to define windows and perform I/O in them.

* New commands which give useful program or environment information have been added. These include LOCKS, PROGRAM DISPLAY and VAR DISPLAY.

* MAX and MIN have been extended to take up to 16 arguments.

* An optional status line is now available via the -S runtime switch.

* The parser now gives diagnostic error messages (this may be suppressed with the -W switch).

* User-defined functions (DEF) have been enhanced to take up to 16 arguments, and can be nested to 26 levels.

* PRINT USING has been enhanced with the addition of a C format to allow centering of strings.

* Quad precision is now the default precision. Runtime switches allow emulation of double and triple precision.

* PACK/UNPACK/RFORM have been enhanced with quad precision format items.

* The VALUE statement has been extended to work with quad precision.

5 Notes and Warnings

5.1 Notes

1) Several terminfo packets are included with Business BASIC for AViiON Systems. These packets are modified versions of DG/UX terminfo packets, and are meant for use with Business BASIC only, though they are quite likely to work correctly with other applications as well. If you experience a problem running another application using a Business BASIC terminfo packet, you should simply cease using that terminfo packet with that application, and use the appropriate terminfo packet supplied with your operating system. You may wish to modify the script which brings up Business BASIC to have it set the TERM environment variable before executing Business BASIC, and reset it after exiting Business BASIC.

   The vt100_bbux and vt220_bbux terminfo packets are meant to be used with Data General terminals running in vt100 or vt220 mode. If you plan to use actual vt100 or vt220 terminals, you may wish to use the system-supplied terminfo packets for those terminals rather than vt100_bbux or vt220_bbux.

   You may modify any terminfo file to fit your specific needs. These man pages contain information which will help you do that: infocmp(1m), terminfo(4), term(5), termio(7), tic(1m), curses(3x), and tty(7).

   To see what modifications were made to the vt220 terminfo packet to produce the vt220_bbux one, execute this command: infocmp -d vt220 vt220_bbux. Similar types of changes may be needed for your non-Data General terminals' terminfo packets for them to run Business BASIC correctly.

2) When Business BASIC is brought up, the default workspace size is approximately 512 kb. Since this is larger than the workspace available under AOS/VS and DG/RDOS Business BASIC, it is possible to create programs which are too large to be ported to those platforms. AViiON BB provides the -p switch which may be used to limit the size of the workspace, thus preventing creation of oversized programs.

   When choosing a -p value, it is important to know that the metacode produced by Business BASIC for AViiON Systems is

larger than that produced by AOS/VS and DG/RDOS Business BASIC. This means that the same source file will produce differently-sized BB save files under AViiON BB and AOS/VS Business BASIC. As a rule of thumb, the AViiON BB file will be about 3 times the size of the AOS/VS one. So a BB program which is 20000 bytes in size according to the AOS/VS BB SIZE command will probably be around 60000 bytes under AViiON BB.

This table shows several -p values, and the approximate corresponding workspace size given under Business BASIC 1.02 by that -p value:

-p Value	Workspace Size
50	24092
100	48668
125	63004
150	75292
200	99868
225	114204
300	151068
400	202268
500	253468
700	355868

Using a switch value of -p500 would give approximately 256 kb, the maximum size possible under AOS/VS BB 5.10.

The -p switch generally does not affect memory usage since the pages making up the 512 kb workspace are not actually used until needed. Thus, if your programs don't use more than 65 kb of the workspace, using -p135 won't allow you to run more users than if you used no -p switch at all.

3) Getting output from a Business BASIC program to a printer is accomplished in AViiON BB by use of a file called DEVICE_MAP, located in SYSLIB. This file allows Business BASIC to handle hard-coded references such as OPEN FILE(15,2),"@LPT". See the section entitled "Managing Devices and Queues" in chapter six of "Using Business BASIC on DG/UX and 386/ix Systems" (093-000685-01) for information of setting up the DEVICE_MAP file. (Note that there is currently a problem which causes the first line of this file to be ignored. Simply repeat your first line when setting up your DEVICE_MAP).

4) Users frequently need to access a serial device from Business BASIC. The DEVICE_MAP file may be used to map hard-coded device names in Business BASIC code to the

appropriate device. However, because of the way UNIX line discipline works, an extra step is needed if the line defaults (300 baud, 8 bit, no parity) are incorrect for the device.

You might assume that stty could be used to set the characteristics, for example: stty 9600 </dev/tty01. However, unless the line is held open by another process like getty, the baud rate will revert to the default at the completion of the stty. The stty causes the line to be opened, and the baud rate to be set to 9600, but when the command terminates, the line is closed. When the open count for a line goes to zero, UNIX systems forget the line discipline settings. This causes problems for users who want to directly access a line without using one of the standard UNIX utilities.

To overcome this problem, a simple C program can be written to open a line, set characteristics using an ioctl() system call, and then hold the line open. A Business BASIC program could then open the line and access the device.

5) One difference between AOS/VS and DG/UX is that PID numbers on DG/UX may be much larger than on AOS/VS. On AOS/VS, the PID is a 2-byte value with the maximum PID value being 65536 (though currently AOS/VS limits this to 4096). On DG/UX the PID is a long integer, and thus can easily grow past a 2-byte value of 65536. Any code which assumes that the PID will be storable as a single-precision number could break if the PID grows beyond this range. For example, this code would return an arithmetic error in line 40 if the PID returned by SYS(9) was greater than 65535:

```
10 DIM A$(10)
20 A$=FILL$(0)
30 PID = SYS(9)
40 PACK "C",A$,PID
```

6) It is possible to find that sysadm does not work after you install a new revision of DG/UX when you already had Business BASIC installed. The reason is that a new ORDER file in the /usr/admin/menu directory was loaded during the DG/UX installation process, and "bbasic_mgmt" (which is added to the file during BB installation) is missing from the file. If this happens, edit the ORDER file to add the phrase "bbasic_mgmt", and sysadm will then work correctly.

7) Note that because of changes to bbux_mgr associated with the new shared I/O implementation, rev. 1.01 obit_limits files are not compatible with rev. 1.02. New obit_limits files should be created for use with 1.02.

8) If Business BASIC is run in the background with output directed to a file, you should also redirect standard error to a file. Otherwise the terminal may appear to hang. The notation 2> is used to redirect error output. For example:

```
bbasic -cd 2> bb.err&
```

9) Here is some information which should clarify the usage of SYS(4): Under DG/UX, a multiplexor is a device. Individual terminals on that multiplexor are units of that device. SYS(4) will return a different value for each unit of that device, so that SYS(4) alone will distinguish between those units. Since the DG/UX operating system provides for a number of devices, it is possible for the unit numbers of different devices to be the same, making SYS(4) unable to provide a distinction between them. In this case, users should also make use of SYS(33), introduced in Business BASIC 1.00. SYS(33) returns the device number.

10) The ESC key is the default primary interrupt key, as in AOS/VS and DG/RDOS Business BASIC. However, due to operating system restrictions, ESC can not be an immediate interrupt key on terminals which generate an escape sequence as part of function key sequences; it is a read-only interrupt key (it will cause an interrupt only when a read is posted). In order to provide an immediate interrupt key, Business BASIC sets the secondary interrupt key to be whatever the DG/UX interrupt key is when Business BASIC is executed. Users may set this DG/UX interrupt by executing an "stty intr <key>" command, where <key> represents the key being designated as the DG/UX interrupt. Again, the DG/UX operating system documentation warns specifically against making the ESC key be the DG/UX interrupt.

11) All SYS function numbers up to 256 which are not documented as being in use are reserved.

12) Business BASIC for AViiON Systems uses two bits of SYS(30). Bit 13 indicates quad precision; its representation is a 4. Bit 12 indicates a UNIX operating system; its representation is an 8.

13) The optional status line (selected by the -S runtime switch) is meant to be used in the development environment only since it imposes some overhead which results in a performance impact. It should not be used when running applications.

14) The Business BASIC CLI expects AOS/VS style pathname templates and switches to be used. It therefore requires pathname conversion be on (the -P switch). It also does

not recognize the pathname exclusion character "\".

Since the CLI requires pathname conversion be on and AOS/VS style templates be used, the use of UNIX file templates will not work with !DUMP, even though the documentation of the !DUMP CLI command states that you may use your operating system file templates. The documentation has been corrected in the documentation file for the Utilities manual which is furnished with revision 1.02.

15) Some users appear to misunderstand the -P pathname conversion switch. The misconception is that it is necessary to use this switch if you are coming from AOS/VS. This is not true. The switch is necessary ONLY if you have hardcoded AOS/VS pathnames, OR if you wish to use the BB CLI. If the filenames in your programs are all simple filenames, then -P is not necessary to run the program; you just have to import the file with an uppercase name.

16) There is a limitation with the QIC-150 tape drive that users should be aware of. This drive hardware can only access 512-byte records. When a 1024-byte record is written, 2 512-byte records are actually written. Therefore if you wish to skip this record later using MTDIO, you would have to skip two 512-byte records.

17) Users who port 2048-byte block index files from pre-5.10 AOS/VS BB revisions must rebuild those index files in order to use the KPREV statement, since the backward links were not maintained in pre-5.10 revisions. If you are porting such files, it is a good idea to rebuild the indexes even if you do not plan to use the KPREV immediately, since INDEXVRFY will now report backward link errors. Otherwise you will receive error messages from INDEXVRFY complaining of problems with the backward pointers.

18) To see what file a Business BASIC link file resolves to, simply cat the file. For example, given these files:

```
.foo
foo
resolution_file
```

where foo is a link to resolution_file, a cat of foo would display "resolution_file".

5.2 Warnings

1) The User Status Table no longer exists in AViiON BB. Much of the information which was kept there is no longer kept, or is kept in different places in different form. Any code which accessed the User Status Table using STMB 5,2,14 or STMB 1,1,14 (or other STMB's) and used information found there will no longer work. Code which used STMB's to store information in memory will not work if the memory locations were calculated based on information obtained from the User Status Table.

2) If ftp is used to transfer files to the UNIX system, it is VERY IMPORTANT THAT BINARY (IMAGE) MODE BE USED!!! If binary mode is not used, the files may be corrupted during the transfer. This corruption is difficult to detect; filesizes may not differ. It is impossible to know afterward whether binary mode was used. Again, if ftp is used to transfer files, USE BINARY MODE!

3) The 150mb cartridge tape drive only supports block sizes that are a multiple of 512 bytes. This is a restriction with this type of drive.

4) Running a D216, D412 or D462 terminal on a mux line in VT100 or VT220 mode with characteristics set to eight bits, 1 stop bit, and no parity will cause display problems on output. Set the terminal to 9600 baud, seven data bits, 1 stop bit, and no parity to avoid these problems.

5) All SYS function numbers up to 256 which are not documented as in use for some purpose are reserved.

6) The Business BASIC reserved word list APERM.PS is now enforced. Programs containing variable names which are on the APERM.PS list must be modified to change the variable names. Attempting to ENTER a LIST file containing reserved APERM.PS items will result in syntax errors for the lines containing those items.

   UNIX offers powerful tools which can help mitigate the impact of this restriction. For example, the AViiON BB BIN directory contains a script which could be used to change occurrences of the variable name LOOP to LOOP1 in a group of LIST files. The script, named make_chng.ex, uses the sed editor to automate the modification process.

7) Attempting to bring up rlsx with a lock table size larger than that you specified in the obit_limits file will fail

with an error reporting that the maximum lock table has been exceeded. Following this error, rlsx won't come up even if you give it the correct lock table size. Your only recourse is to bring down obit, then bring up obit, and then bring up rlsx with the correct lock table size.

8) When you bring down rlsx, you should bring down obit. Attempting to bounce rlsx (bring it down and then back up) is likely to result in the following warning message from obit when rlsx is brought down:

```
obit: WARNING - unable to unlock rlsx lock table from memory
```

If this problem occurs, this message will be sent to the obit output file or the console, depending on your choice when bringing up obit. However, an "OK" message is given in the Daemon Management menu.

A subsequent attempt to bring up rlsx will then provoke this message from rlsx:

```
rlsx: ERROR - unable to lock lock table in memory
```

If these events occur, obit must be brought down before rlsx will come up.

9) To delete a link, the user must use the BASIC CLI command UNLINK or STMC 35. In AViiON BB the DELETE statement or command deletes the resolution file only, not the link file. This causes a situation in which a link file references a non-existent resolution file. Business BASIC returns the error "Cannot access {link_name}" when references to this type of file are made.

Note that this situation is the same as under AOS/VS and DG/RDOS BB, where using DELETE to delete a link file deletes the resolution file.

To create a link, use either the BASIC CLI command LINK or STMC 21.

10) Due to the way that the Business BASIC parser works, a situation exists with the VAR RENAME command which may cause confusion. If the command VAR RENAME A,B is typed and variable A does not exist, no error message is issued. This happens because when the command is parsed, an entry for A is created. This is long-time Business BASIC functionality; typing LTE B=C instead of LET B=C adds variable LTE to the program.

These "bogus" variables may be removed by LISTing the

program to a disk file and then ENTERing it.

11) Since the CSM utility is function-key driven, it can only run on terminals where function keys are supported. Since the vt100_bbux terminfo packet does not support function keys, CSM may not be run on a terminal which requires this packet.

12) Some AOS/VS Business BASIC programs depend on the AOS/VS characteristic FKT (treat function keys as delimiters) being turned ON. These programs will not work the same under Business BASIC for AViiON Systems, since there is no corresponding facility in the DG/UX operating system. Adding an STMA 4,4,SYS(50) or STMA 4,5,SYS(50) to the program will cause the primary or secondary unpend key respectively to be the function key header; inputs will then unpend on any function key. As an alternative, PRINT @(-5,SYS(50)) or PRINT @(-4,SYS(50)) can be used to do the same thing. Under Business BASIC 1.02, nothing is echoed in this case.

13) The contents of the buffer string used by the index file routines differ from the index buffer string in AOS/VS and DG/RDOS Business BASIC. USERS ARE AGAIN REMINDED THAT PROGRAMS SHOULD NOT MAKE USE OF THE CONTENTS OF THIS STRING SINCE IT IS SUBJECT TO CHANGE WITH ANY REVISION!!!!!

14) Depending on how much memory is available, there is a point where attempting to bring up another Business BASIC process will fail because there is not enough memory available to support another process. In this event, Business BASIC may display one of several different error messages: "Fatal initialization error - Operating system initialization", "Fatal initialization error - User internals initialization" or "Fatal initialization error - System internals initialization."

15) PRINT "<7>" when -C switch has been specified may not always ring the bell on a D412 terminal. This is corrected by FCO # 900622. If under warranty, call 1-800-DG-HELPS and explain that you are encountering a problem fixed by this FCO, and that you need to return the logic board under warranty. If under service contract, contact your local Data General office to arrange a swap of the logic board.

16) Running Business BASIC through telnet may cause SYS(10) to be set to 1 when Business BASIC comes up, due to the interaction of curses and telnet.

17) SYS(9) returns the process id (PID). Under AOS/VS, this number has tended to be in a range from one digit to three

digits (at one time the maximum allowable PID was 64 on AOS and 255 on AOS/VS). Under UNIX, this number frequently is in a five-digit range. If a program has been coded in such a way that SYS(9) is expected to be three digits or less, the program may not work correctly.

18) In CSM, the arrow keys and home keys do not work. A way to position to a previous character or line is to hit newline until the cursor wraps back to the top of the screen.

19) The first line of the DEVICE_MAP is ignored. You must put a dummy entry on the first line so that you can access the device listed on the second line.

20) "I/O Error 4 - Not a save file" is returned when you try to LOAD, RUN, CHAIN, or SWAP to an incompatible save file.

21) An I/O error is returned from MTDIO if you attempt to skip backward at the beginning of the tape.

22) In bbux_mgr (or daemon_mgmt via sysadm), from any ">" prompt if you go to help and return, then you no longer have the ">" prompt and a full line to enter the response. Instead, the cursor is on the same line as the prompt text. If you enter a response that wraps to the next line, you cannot get back to the first line to correct typing mistakes.

23) An "obit already exists" error goes to the system console instead of the terminal if you are attempting to bring up obit when it is already up and the terminal you are using is not the master console.

24) Executing Business BASIC and redirecting the output, then executing a TRACE with an output filename can cause unprintable characters to appear in the TRACE output file after the time stamp.

25) STME 10, which retrieves the pathname of a file opened on a specified channel, will cause a core dump if the pathname is not a file or directory but a device or queue.

26) The TRACE filename is lost when LISTing a statement like: 10 TRACE ON,"foo". Such a statement may be ENTERed and runs correctly.

27) A bogus "Error 53 - Renumbering error(s)" occurs when a program containing a block IF is renumbered. The renumbering is done correctly.

28) RENUMBER renumbers some but not all line numbers in an ON GOTO/GOSUB statement without returning an error.

29) In programs containing an active DO loop, PROGRAM DISPLAY is not correctly displaying the line number containing the control expression.

30) An error that occurs in an IF does not set SYS(20) or return the line number where the error occurred when reporting the error. For example, executing this line:

```
20 IF A THEN LET X=1/0
```

would give the message "Error 16 at 0 - Arithmetic"; SYS(20) would contain a zero.

31) If Business BASIC users open files in shared mode, then log off Business BASIC without closing the files, obit may produce a message something like "Flushing pages of foo" (where foo is the file opened in shared mode but not closed before the BB user logged off). The obit daemon does not flush pages which have not been modified, so even though obit's message indicates that pages are being flushed, the only ones which really are flushed are modified pages.

32) The config_mgmt script recognizes invalid responses as "yes" for all six y/n prompts.

33) During installation, an invalid response to the question, "Do you want the Business BASIC daemons to start automatically at system init time? (y,n) [y]:" is taken as a yes.

34) PRINT FILE returns a "line too long" error if an attempt is made to print a line of 132 characters. This does not happen with PRINT, just PRINT FILE.

35) STMA 14,X$,2 does not correctly convert EBCDIC to ASCII.

36) INPUT FILE and INPUT FILE USING should return error 18, "line too long", when doing input from a file containing more than 132 characters before a delimiter is found. Currently no error is returned, and 133 characters can be input. When 133 characters are input, the delimiter string will not contain the delimiter.

5.3 Using UCALLs

Business BASIC for AViiON continues to offer users an interface to user-written subroutines through the UCALL statement.

Previous release notices for Business BASIC for AViiON Systems have contained a note stating that the UCALL statement was under consideration for removal in a future revision. Following due consideration, it has been decided that the UCALL statement will not be removed from the language and will be retained in future revisions for the sake of existing code and applications.

Users should be aware that a more complete C language interface is planned for a future revision. Implementors of new code will be strongly encouraged to make use of the new interface instead of the UCALL one.

To make use of the UCALL interface, the user-written subroutines must be written in the "C" langauge, not assembly language. Any subroutine written in 16- or 32-bit assembly langauge will need to be recoded. Subroutines compiled by the following "C" language compilers are supported:

Business BASIC for DASHER/386	Green Hills C, PCC
Business BASIC for AViiON Systems	GNU C

An example of a C subroutine is included in e_ucl_usubs.c. It is recommended that you study this example as you read this documentation.

There are two ways you can implement your UCALLs. You can either put your routines directly in e_ucl_usubs.c, or you can place them in separate files.

If you place them in separate files:

1) Include the file ucall.h in your UCALL source files. This file resides in BASICGEN/lb. A copy of it should be placed in /usr/include.

2) Compile the UCALL source files with a supported compiler.

3) Archive the UCALL object modules in ucl.a using ar(1) as described in the UNIX Programmers Reference Manual.

Whether or not you have your routines in e_ucl_usubs.c or in separate files, next:

1) Add the names of subroutines, their maximum and minimum argument counts, and argument types to the ucl_ucall_entry table in e_ucl_usubs.c In the same file, assign the total number of ucalls to ucl_allocated_max_ucalls.

2) Compile e_ucl_usubs.c with a supported compiler.

3) Use bbasic_build to produce a new interpreter.

The library ucl.a and the module e_ucl_usubs.o are automatically loaded into any Business BASIC interpreter. The default ucl.a supplied on your release media defines no routines.

A UCALL can have up to eight arguments. Each UCALL which you will want to use must have an entry in e_ucl_usubs.c. The structure of the ucl_ucall_entry table in e_ucl_usubs.c is:

```
typedef struct
    {
    int                         max_args;
    int                         min_args;
    ucl_argument_enum_type      arg_types[8];
    void                        (*function)();
    } ucl_ucall_type;
```

In addition, you must indicate the maximum number of ucalls you will use in an interpreter by defining ucl_allocated_max_ucalls to be this value in the file e_ucl_usubs.c.

The interpreter will use the number supplied as an argument to the UCALL statement in a Business BASIC program as an index into a table of entries of the type ucl_ucall_type and will make sure that the number of arguments given is within the range specified to that UCALL entry. The interpreter will evaluate each of the arguments, then call the function whose address was specified in the given entry.

Your "C" subroutine may use the arguments to the UCALL statement in a Business BASIC statement by including the header file ucall.h, in /usr/include. The ucall.h file contains definitions which allow you to examine and modify the value of these arguments. The arguments are placed on an internal stack which is made up of entries which have the following structure:

```
typedef struct
 {
   ucl_operand_enum_type                    kind;
   union
     {
       ucl_string_operand_type              string;
       ucl_string_literal_operand_type      literal;
       ucl_numeric_ref_operand_type         numeric_ref;
       ucl_numeric_temp_operand_type        numeric_temp;
     } op;
 } ucl_expr_stack_operand_type;
```

The elements of this structure are also defined in ucall.h. In addition, the variable ucl_get_ucall_arg1_pointer is defined. This

variable is a pointer to a stack operand which corresponds to the first argument in the UCALL statement. Since arguments are pushed on the stack in order, the operand on top of the stack will be the last argument in the UCALL statement. Your routine can find the number of arguments supplied to the UCALL statement in the variable ucl_number_of_args_passed.

Several restrictions apply to the argument operands. Your "C" subroutine may not alter the contents of the numeric_temp or literal operands. Also, your routine must not alter the argument stack. The contents of variables, both string and numeric, may be altered by changing the values which are located at ucl_get_ucall_arg1_pointer->op.string.string_ptr and ucl_get_ucall_arg1_pointer->op.numeric_ref.ref_storage_ptr. Keep in mind, however, that your routine must not overwrite descriptor fields in the variable storage area and that numeric values are always stored in big-endian representation. Two macros are supplied in ucall.h which may be used in converting numeric values from little-endian to big-endian, or vice versa. They are named UCL_BIG_ENDIAN_TO_TARGET and UCL_TARGET_TO_BIG_ENDIAN.

If your subroutine performs any output to the terminal, unpredictable results may occur after returning from the routine. The interpreter does not use the "C" language input and output routines defined in stdio.h. Because of the difference, it is recommended that your subroutine not perform output to the terminal.

Your subroutine may raise an error by calling the routine ucl_ucall_error and passing it an integer which is the Business BASIC error to be raised.

5.4 What to Do if Business BASIC Terminates Abnormally

1) Potential Business BASIC abnormal termination problems fall into these categories:

   * An obit or rlsx process reports a bad status
   * One or more Business BASIC processes hangs

   If obit or rlsx reports a bad status, do the following:

   1) First select option 11 in bbux_mgr to produce a debug dump file. This could be very useful in determining the cause of the problem.
   2) Log off any Business BASIC users

3) If obit is executing normally and rlsx is not, use option 5 of bbux_mgr to stop rlsx, and then option 4 to restart it. Use option 6 to ensure rlsx is now running normally. If it is not, go to the next step.

4) If the obit process reports a bad status, or rlsx is still reporting a bad status after being restarted, take these steps:

   a) Use option 5 of bbux_mgr to stop rlsx, then option 2 to stop obit. It is important that you take down these processes in the order indicated.

   b) Use option 1 to start obit and then option 4 to restart rlsx. These steps must be performed in the indicated order.

   c) Check the status of both processes. If both are not running normally, proceed to the next step.

5) If you performed step 4 and both of the processes are not running normally, reboot the operating system and restart Business BASIC.

If one or more BB processes hangs, follow these steps:

1) First select option 11 in bbux_mgr to produce a debug dump file.

2) Execute an ipcs command, redirecting the output to a file (e.g., ipcs > out.file).

3) Execute a ps command with options of -ef, and redirect the output to a file (e.g., ps -ef > ps.out). Wait a short while and repeat the process, specifying another output file.

4) Execute a sysdef command and direct its output to a file. All of these files will be very useful in determining the cause of the problem; they should be sent when reporting the problem.

5) From the bbux_mgr menu, execute the option which flushes shared pages.

6) Kill the hung process or processes. Log off any other Business BASIC users.

7) Use options 5 and then 2 to stop first rlsx and then obit. After both are stopped, use options 1 and 4 to bring up first obit and then rlsx.

8) If you cannot stop obit, or if both obit and rlsx do not restart and run normally, reboot the operating system and restart Business BASIC.

2) Terminal characteristics may have been left in an unknown state if Business BASIC was terminated abnormally but UNIX did not crash. To remedy this, try typing a ctrl-m or ctrl-j followed by an exit command terminated by a ctrl-m or a ctrl-j; even though it may not echo on the screen, it may get executed, which will cause the getty process monitoring the line to be reinitialized. It may also be possible to run a script which executes stty commands which will reset the characteristics. You can also terminate the getty process from the root console so that the getty process for the line will be reinitialized.

3) If any index files were open when the abnormal termination occurred, they may have been damaged. The files should be examined with INDEXVRFY before being used. Non-index files may also have been damaged; they should be examined using whatever methods (utilities, etc.) your application may include.

5.5 Language Elements Under Consideration for Future Removal

1) The BB CLI is furnished in Business BASIC for AViiON Systems as a means of easing the transition from AOS/VS or DGRDOS to the DG/UX operating system. However, part of the BB CLI is under consideration for removal in a future revision.

   It is anticipated that in a future revision a better, more generic method will be provided for doing the types of work done from the CLI. At that point the CLI commands themselves will no longer be part of the product. Business BASIC will retain the capability of passing switches and arguments to a program by putting a string into the common area and executing a SWAP"CLI.

6 Documentation

6.1 Manuals

Part Number	Title
093-000684-00	Learning Business BASIC
093-000351-01	Commands, Statements, and Functions in Business BASIC
093-000389-01	Subroutines, Utilities, and the Business BASIC CLI
093-000685-01	Using Business BASIC on DG/UX and 386/ix Systems
085-600140-02	Business BASIC for AViiON Systems Release Notice

6.2 Documentation Update Files

Documentation update files for three manuals are provided in the Business BASIC DOC directory. They are:

* u093000351.01 - Updates the "Commands, Statements and Functions" manual
* u093000389.01 - Updates the "Subroutines, Utilities and Business BASIC CLI Commands" manual
* u093000685.01 - Updates the "Using Business BASIC on DG/UX and 386/ix Systems" manual

7 Software

7.1 Media

Model Number	Part Number	Description
L005ASU	079-600029-02	QIC 150 Cartridge Tape

7.2 Organization

The cartridge tape has the following layout:

File	Name	Type	Disposition
0	reserved	image	
1	sysadm_toc	toc	
2	reserved	image	
3	reserved	image	
4	BBUX_usr.img	tar	/usr required movable
5	BBUX_uopt.img	tar	/usr/opt optional movable

7.3 Files

A complete list of the files comprising this release is contained in bb88k_1.02.fl

8 Installation Instructions

8.1 Loading from Tape

1) Release contents:

The supplied tape contains the following directory structure and files:

BASICGEN - directory for generation of Business BASIC interpreters
BASICGEN/lb - directory containing files needed to generate a Business BASIC interpreter
SYSLIB - directory containing system utilities & libraries, as well as BASIC.ER and DEVICE_MAP
BIN - directory containing various executables, including the supplied Business BASIC interpreter. This interpreter is a development system which uses in-memory SWAPs and allows all users to use privileged statements.
BIN/bbux_mgr - script for managing obit and rlsx
BIN/rlsx - resource locker daemon
BIN/obit - daemon which manages cleanup after bbasic terminates
BIN/bbasic - a default interpreter
BIN/profile.ex - an example user profile script
BIN/run_bbasic.ex - sample script for invoking BB
TERM - Business BASIC terminfo directory
TERM/v - contains vt220_bbux and vt100_bbux, modified terminfo files
TERM/a - contains at386_bbux, a modified terminfo file
TERM/A - contains AT386_bbux, a link to at386_bbux
TERM/d - contains modified terminfo files, including d216+_bbux, d412+_bbux, and d462+_bbux, d210_bbux, d211_bbux, d214_bbux, d215_bbux, d410_bbux, d411_bbux, d460_bbux, d461_bbux
TERM/p - contains port_bbux, a modified terminfo file
CONVERT - directory containing porting aids
CONVERT/li.a - update module for BB 1.01 which corrects a LIST bug which could hinder
DOC - directory containing selected utility sources and documentation files. The release notice, 085600140_02, is found here.
DOC/bb88k_1.02.fl - file containing a list of all files in the release
/usr/options/bbux.name - used by sysadm
/usr/admin/menu/bbasic_mgmt - directory used by sysadm.

```
        Contains scripts to manage Business BASIC
        including daemon_mgmt, DESC, config_mgmt,
        init_options, and run_bb.
/etc/rc2.d/S392.bbasic - BB script for system initialization
/etc/rc0.d/K392.bbasic - BB script for system termination
/etc/rc1.d/K392.bbasic - BB script for system termination
/etc/rc5.d/K392.bbasic - BB script for system termination
/etc/rc6.d/K392.bbasic - BB script for system termination
/etc/rcS.d/K392.bbasic - BB script for system termination
```

The last six files listed above (S392.bbasic and K392.bbasic) will be present on the system only if you choose to have the Business BASIC daemons start automatically. This decision is made either during the installation procedure, or through the sysadm bbasic_mgmt menu's new init_options selection.

2) Installing the product:

This revision of AViiON BB requires a save file revision. Prior to installing it, you will need to have LIST files of your programs available.

A module named li.a is furnished in the CONVERT directory which corrects a problem in LIST in revision 1.01 of Business BASIC. This problem would cause a core dump if an attempt was made to LIST to a filename, and the program being listed had GOTOs or GOSUBs to non-existant line numbers with the GOTOs or GOSUBs located inside either FOR/NEXT, DO/WHILE or block IF loops. If you are converting from rev. 1.01, you should first install this module in 1.01 by putting it in the 1.01 BASICGEN/lb directory, and then building a new 1.01 Business BASIC interpreter by use of the build_bbasic script. This new interpreter may then be used to produce the needed LIST files.

If you have an earlier version of AViiON BB , make sure that its obit and rlsx are down.

Under DG/UX, it is required that Business BASIC be loaded into a file system named /usr/opt/bbux. This file system should be created and mounted before loading the tape. Use the sysadm diskmgmt command to create a logical disk and file system for Business BASIC. Then use the sysadm addfsys command to make the file system accessible, specifying /usr/opt/bbux as the mount point directory. (See "Installing and Managing the DG/UX System", 093-701052-02, for information on these procedures). The release itself takes approximately 15K blocks. To allow for additional interpreters and user programs, you may want to allocate 18 to 25K blocks for the release.

If your system contains an earlier version of Business BASIC, and you plan to install the new version in the directory

containing that earlier version, backup the earlier revision before you delete it. If there are files you have added to the Business BASIC directories which you will want to place in the new BB directories, move them to some temporary location before deleting the earlier version. After the new version is loaded, you can move those files back.

A file containing a list of all the files in the release is provided with each Business BASIC revision. Its filename looks something like bb88k_1.02.fl. This file may be used to locate and delete all the parts of the earlier release.

Log on as root on the console in order to load the media.

Insert the tape into the tape drive.

Business BASIC is installed using the sysadm facility. Simply log on as root, and type:

```
sysadm
```

This causes the sysadm main menu to be displayed. Select the "releasemgmt" option. When the Software Release Management menu is displayed, select the "loadpackage" option. It will take you through the following dialogue. Take the defaults when offered.

```
Release Area [PRIMARY]: (you hit return)
Tape Drive [0]                (you hit return)
Is the tape mounted and ready? (you type y)
Load package BBUX [yes]  (you hit return)
List filenames while loading? [yes]  (you hit return)
Mount volume 1
Is the tape mounted and ready?  (you type y)
```

At this point, the loading begins. The message, "Skipping tape files 0 to 3" will be displayed. Then the message "Loading file BBUX__opt.img" is displayed, followed by the filenames making up the Business BASIC release. When all the files have been loaded, the message "loadpackage is finished" is shown. Press the NEWLINE to return to the releasemgmt menu; at that menu, select the "setuppackage" option. The dialogue will look something like this:

```
Release area? [PRIMARY]
The following packages have setup scripts that have not been
run:

    bbux
    (possibly other names)
```

```
Package name? [all]  (type bbux)
Processing setup scripts for package bbux.
Setup package bbux in usr? [yes]  (type return)

            Setting up Business BASIC

Do you want the Business BASIC daemons to start automatically
at system init time? (y,n) [y]:
```

Having the daemons start automatically spares the system manager from having to remember to do this manually every time the system is brought up, and ensures that the daemons will always be available for Business BASIC's use. However, if you want, you may say "no" at this point. If you later decide that you want to change your decision on this question, you may do so through the sysadm bbasic_mgmt menu.'

NOTE: Business BASIC 1.02 requires that the obit daemon be running; if it is not running, a fatal initialization error will occur when trying to bring up the interpreter. It is not required that rlsx be running; however, if it is not, RLSX_DIR must not be set, or a fatal initialization error will occur when trying to bring up BB.

If you ask for the daemons to start automatically, you are asked six additional questions to determine where the daemons will run, where the obit daemon will find its operating parameters, and the size of the rlsx lock table:

```
Enter the directory for obit [/usr/opt/bbux/BIN]
>
Enter obit limits filename [/usr/opt/bbux/BIN/obit_limits]
>
Enter pathname for messages from obit [obit.msg]
>
Enter the directory for rlsx [/usr/opt/bbux/BIN]
>
Enter the lock table size in bytes [8192]
>
Enter pathname for messages from rlsx [rlsx.msg]
>
```

You are next asked if you wish to generate a Business BASIC interpreter now. This does not have to be done now, since there are ways to do this as described in "Using Business BASIC on DG/UX and 386/ix Systems" (093-000685-01). If you want to generate one now, please consult that manual for more details.

After the question about generating a Business BASIC interpreter has been answered, a message appears stating that Setup-package is finished.

Business BASIC is now loaded on your system. Exit sysadm by typing q at each prompt until you are returned to the shell.

8.2 Reconfiguring the Rev. 4.30 DG/UX Kernel

Note: These instructions are based on revision 4.30 of DG/UX. Later revisions of DG/UX may vary. If you have a later revision of DG/UX, you should also consult the release notice which accompanies it. You may also wish to consult your operating system documentation.

Certain tunable parameters in the UNIX kernel must be modified in order for Business BASIC to work correctly. This is because the shared memory and locking schemes depend on these parameters being modified. There are other tunable parameters which should be modified if you want to increase the maximum open files per process above 60. The Business BASIC Daemon Management Menu, available through sysadm, offers new options to aid in the process of modifying these parameters. Option 7 allows you to create files describing your operating environment, while option 8 uses the information in those files to calculate and display recommended values for the tunable parameters which must be changed to satisfy those requirements.

The steps you will follow are:

1) Determine the requirements of your Business BASIC applications.

2) Use option 7 to define your requirements for obit.

3) Use option 8 to determine kernel configuration requirements for your operating environment.

4) Reconfigure and install the kernel.

5) Boot the new kernel.

6) Bring up obit and rlsx (if they were not brought up automatically at system boot time).

When you choose option 7, you are asked first for the name of the output file, with obit_limits being the default. You may choose another name if you like. (If you choose the default or specify a simple filename, the file is created in the bbux/BIN directory. You can create the file in a different directory by specifying a full or partial pathname.) You then are asked these questions (defaults are in brackets):

```
Enter maximum number of rlsx's [2]:
Enter maximum number of bbux's [10]:
Enter maximum number of open shared files on the system [20]:
Enter maximum lock table size in bytes [8192]:
Enter maximum number of 2048 shared file page buffers[300]:
Enter maximum number of ipc files[0]:
Enter maximum number of buffered STME messages per user[0]:
Enter maximum message length of an STME message[0]:
```

The first two questions ask you to specify the maximum number of rlsx lock servers which may be running at any one time, and the maximum number of Business BASIC users which may be logged onto Business BASIC at any one time.

The third question asks for the maximum number of files which may be open in shared mode at any one time. Answering this question requires some knowledge of the Business BASIC application(s) which will be running on the machine. If the number supplied here is too small, a Business BASIC program which tries to open a file in shared mode will receive error 108, "Maximum number of open shared files exceeded."

The fourth question asks for the maximum lock table size. This refers to the largest lock table to be used by any rlsx process. AViiON BB allows multiple rlsx processes to be running at one time; each rlsx will have its own lock table. The size of the lock table is specified when the rlsx process is brought up. The value given for this question should be equal to or greater than the size of the largest rlsx lock table. If during installation you requested that rlsx be automatically started at system boot time, then your response to the fourth question should be equal to or greater than the lock table size you gave during installation, unless you will be running a second rlsx with a larger lock table size. This formula may be used to calculate the size of the lock table:

```
size = 8 + ((max_filename_len>+45) * max_locks )
```

where max_filename_len is the maximum length of any filename that will be used in a LOCK statement and max_locks is the maximum number of locks that will need to be in effect at any one time. So for example, if your programs had a maximum filename length of 10 used in a LOCK, and you could have 100 locks in effect at any one time, the size would be 8+(100*50), or 5008 bytes. In this case the default of 8192 would suffice. Also, note that when this formula is used, the lock table will hold at least max_locks number of locks. Since each unique filename is stored only once, you will be able to accommodate more than max_locks locks if more than one LOCK is made with the same filename.

The fifth question asks for the maximum number of 2048 shared file page buffers. Note that this does not have anything to do with the

size of shared files. The size of shared files no longer has any bearing on shared I/O support. What this question refers to is the number of shared file pages which can be in obit's shared memory segment at any one time. The value specified can be very important to shared I/O performance. If the value is too small, excessive flushing of buffers will occur in order to make room for new pages. If the value is too large, extra memory will be unnecessarily reserved for obit's use, and excessive DG/UX paging activity may take place. It is a good idea to take the default offered here, and then use the new bbux_mgr option which allows you to monitor paging activity to determine whether you should change the value. There is a discussion in the Enhancements section about this new option; please refer to that discussion.

The sixth question asks for the maximum number of ipc files which may be open at any one time. An ipc file is opened only via an STME 23. If the maximum number of ipc files are already open and an STME 23 is executed, a Business BASIC error 111 will occur, "Maximum number of open ipc files exceeded." If your programs do not make use of STME's 20-26 to perform interprocess communications, you should take the default of zero for this question.

The seventh question asks for the maximum number of STME messages per user at any one time. For example, if a user creates two ipc files, any messages sent to the global ports associated with either of these ipc files is sent to the message queue of this user. If the maximum number of spooled messages is exceeded, Business BASIC error 112, "Maximum number of spooled messages exceeded" will occur.

If your programs do not make use of STME's 20-26 to perform interprocess communications, you should take the default of zero for this question.

The eighth question asks for the maximum message length (in bytes) of an STME message. If a user attempts to send a message of a longer length, a Business BASIC error 113 will occur, "Maximum message length exceeded." Your answer to this question should be the actual greatest length your STME messages will be plus one.

If your programs do not make use of STME's 20-26 to perform interprocess communications, you should take the default of zero for this question.

After you have provided the requested information, the file containing your responses is placed in the directory you specified. You should now select option 8 to calculate and display the tunable parameters which must be modified, and the values needed. Option 8 first prompts you for the name of the file you created with option 7 (or lets you choose to create that file). It then displays the

desired information. You can choose to have this placed in a file rather than on the screen.

The display has a screen with four columns headed "PARAMETER", "OBIT", "DEFAULT", and "CHANGE", with the tunable parameters listed in the first column. In the "OBIT" column is the value which the obit process requires to operate correctly based on the limits supplied by the user. The "DEFAULT" column contains the DG/UX default value for that parameter. The "CHANGE" column indicates that you should make a change for that parameter; the indicated value is a minimal recommendation based only on obit's needs. If there are other processes which use the system resources affected by these parameters, the parameter settings may have to be higher.

Once you have the information provided by option 8, you are ready to reconfigure the kernel. This should be done before beginning to use Business BASIC.

Once the kernel is reconfigured, the system must be rebooted in order for the changes to take effect.

To reconfigure the kernel, logon as root, and type:

```
sysadm newdgux
```

You will be prompted for the name for your new system. If it does not exist, you will be asked if it should be created.

Next you are prompted for the name of an editor, with "vi" supplied as a default. The reason for this is that you will be editing a system file to modify the tunable parameters. Select the editor you prefer.

You will find yourself editing a file in /usr/src/uts/aviion/Build called system.systemname, where systemname is the name you provided for your system. You need to edit a section called "Tuneable Configuration Parameters" for DG/UX. There are three such sections in this file; you want to edit the one corresponding to DG/UX (the other two are for NFS and TCP/IP). The DG/UX one appears first in the file.

For each parameter, simply add a line to the "Tuneable Configuration Parameters" section for DG/UX. Each line should look something like this:

```
SEMMNI          56
```

Exit the editor. (Note: it is possible that you will receive an error message from sysadm when you leave the editor. If this happens, you will be told to answer the next question "NO" which will cause you to be back in the editor, editing the same file. In

this event, check to be sure your changes indeed are in the file.)

You will now be prompted "Ready to Configure a Kernel? [yes]". If you answer no, you are prompted for an editor name again, and then placed back in the edit session. If you answer yes, you see a message like "sysadm will now run config on /usr/src/uts/aviion/Build/system.systemname", where "systemname" is the name you chose. If there are no problems, you will see the message "Config succeeded", followed by "sysadm will now attempt to build a kernel. Building...". If the build succeeds, the message "The build succeeded" is displayed, followed by the prompt "Install the New Kernel? [no]". Installing the kernel does not shut the system down; it merely moves some files and renames some others. The new kernel does not take effect until the next time you reboot the system.

If you do not choose to install the kernel at this time, responding "no" ends the dialogue with a message confirming that the kernel has not been installed, and informing you that the file is in /usr/src/uts/aviion/Build.

If config has problems, they may be caused by a typographical mistake you made when editing the system file. Try the procedure again, using the same system name, and make certain you have not typed a parameter name incorrectly.

8.3 Generating a Business BASIC Interpreter

The default interpreter which is supplied with Business BASIC Rev. 1.02 is a development system which uses the new SWAP implementation and allows all users to use the privileged statements. If this meets your requirements, you do not have to generate a new interpreter. Should you wish to generate a new interpreter, please consult the "Using Business BASIC on DG/UX and 386/ix Systems" (093-000685-01) manual.

8.4 Managing the Business BASIC System

Once Business BASIC is installed on your computer, the obit and rlsx processes must be started in order for Business BASIC to run correctly. Please see the "Using Business BASIC on DG/UX and 386/ix Systems" (093-000685-01) manual for information on starting and managing those processes.

If you chose for Business BASIC daemons to be automatically brought up when the system is brought up, and if you leave the system running full-time, then you should not ordinarily need to start those processes.

obit and rlsx are meant to be started and stopped by the scripts provided. If for some reason you want to write some scripts of your own which start and stop those daemons, be sure to use the same command lines used by the Business BASIC scripts.

8.5 Managing Devices and Queues

Business BASIC programs may direct output either directly to a device or have the output queued through the lp spooler. To accomplish this, some initial setup work is required. This work is described in the manual entitled "Using Business BASIC on DG/UX and 386/ix Systems" (093-000685-01).

8.6 The Business BASIC Execution Environment

The Business BASIC execution environment requires that a number of environment variables be set up prior to running Business BASIC. Setting up the execution environment is discussed in the manual "Using Business BASIC on DG/UX and 386/ix Systems" (093-000685-01). This manual also details the command line options available with BB 1.02.

8.7 Importing Business BASIC Programs

See the "Using Business BASIC on DG/UX and 386/ix Systems" (093-000685-01) manual for a discussion of importing Business BASIC programs to UNIX.

9 Volume Label File Compatibility

Revision 4.20 of AOS/VS Business BASIC introduced a new format for the volume label (.VL) file; revision 1.02 continues to use that new format. Thus the volume label file produced in revision 1.02 is not compatible with pre-4.20 revisions of AOS/VS Business BASIC or with pre-8.20 revisions of ECLIPSE RDOS or DG/RDOS Business BASIC.

A conversion procedure for your existing volume label files will be necessary if you are revving up to Business BASIC for AViiON Systems from a pre-4.20 revision under AOS/VS or a pre-8.20 revision under DG/RDOS. A utility program named VLCONVERT.BA is included with this release to facilitate this conversion. This program is intended to be run on your AOS/VS or DG/RDOS system to convert the volume label files before moving them to UNIX. The program will reformat each logical entry (record) in each volume label file produced under a previous revision, and convert each entry into the new volume label structure.

Comments in the VLCONVERT.BA file describe how it is used.

Note that database (.DB) files need not be changed - only the volume label (.VL) files.

10 Preparing a Software Trouble Report

Should you encounter a problem in this revision of Business BASIC, report it to Data General on a Software Trouble Report (STR) form. Please observe the following guidelines when filling out this form.

NOTE: A very common mistake is to import data files using ftp without using the binary option. This can cause all sorts of problems. Be very certain that data files have been properly imported before filing an STR.

1) The STR should contain only one problem or suggestion to facilitate its classification and processing.

2) The problem should be clearly stated so that it is possible for support personnel to reproduce it. Vague statements such as "it doesn't work right" should be avoided. If an error message is displayed, the exact message should be recorded on the STR. If possible, include the error numbers returned from SYS(20), SYS(40), SYS(41), SYS(42) and SYS(43).

3) If possible, supply a BRIEF program or a procedure that reproduces the problem. If the program extends beyond ten lines, it should be sent in machine readable format. Please send only the program(s) and file(s) needed to reproduce the problem. It is generally not necessary to send a copy of your entire system.

4) If sample programs require a database, some or all of the database must be supplied. If this is not possible, please include details of how to construct a database that can be used to reproduce the problem.

5) The Business BASIC sysgen log file should be included. If it is available, include the /usr/src/uts/aviion/Build file for your operating system. It will have a filename similar to "system.yoursys" where "yoursys" is the name of the kernel you reconfigured.

   Execute the sysdef command and redirect its output to a file; include that file.

6) The method used to bring up Business BASIC should be described, including the switches used.

7) Include full information concerning the particular hardware configuration upon which the problem occurs. For example, it is not sufficient to report that "Business BASIC doesn't PRINT correctly to the printer." It would be necessary to note

specifically the type of printer, its switch settings, how it was being accessed, and exactly what problem was occurring. If the problem involves the display on the CRT, include the type of terminal.

8) If the problem being reported is an ERROR 4, the Business BASIC sysgen log file should be enclosed. The exact message of the ERROR 4 should be recorded on the STR, including all numbers displayed and the program that was running when the ERROR 4 occurred.

9) Any STR should always indicate the revision of Business BASIC and the revision of the operating system. If the problem occurred soon after installing a new revision of the operating system and/or Business BASIC, or new hardware, please note this on the STR.

10) Suggestions should be clearly described. In addition, the STR should indicate the need for implementation of the suggestion.

11) Note that additional error checking may affect performance. Business BASIC attempts to check for errors and conditions that might be reasonably expected to occur, but does not always check every possible condition in order to avoid undesirable impact on performance. Please bear this in mind if submitting a request for additional error checking.

12) Anything sent in machine readable format should be sent on QIC 150 cartridge tape, 5.25" 1.2 mb diskettes, or 3.5" 1.4 mb diskettes. The media should be clearly labeled stating contents, format, date, etc. If possible, place more than one copy of each file on tape unless the files are very large. For example, if file 0 has a dozen files necessary for reproducing the problem, place those same files also on file 1.

13) No programs which have been protected should be submitted unless the problem does not occur in an unprotected version. In that case, submit both a protected and an unprotected version.

--End of Release Notice--