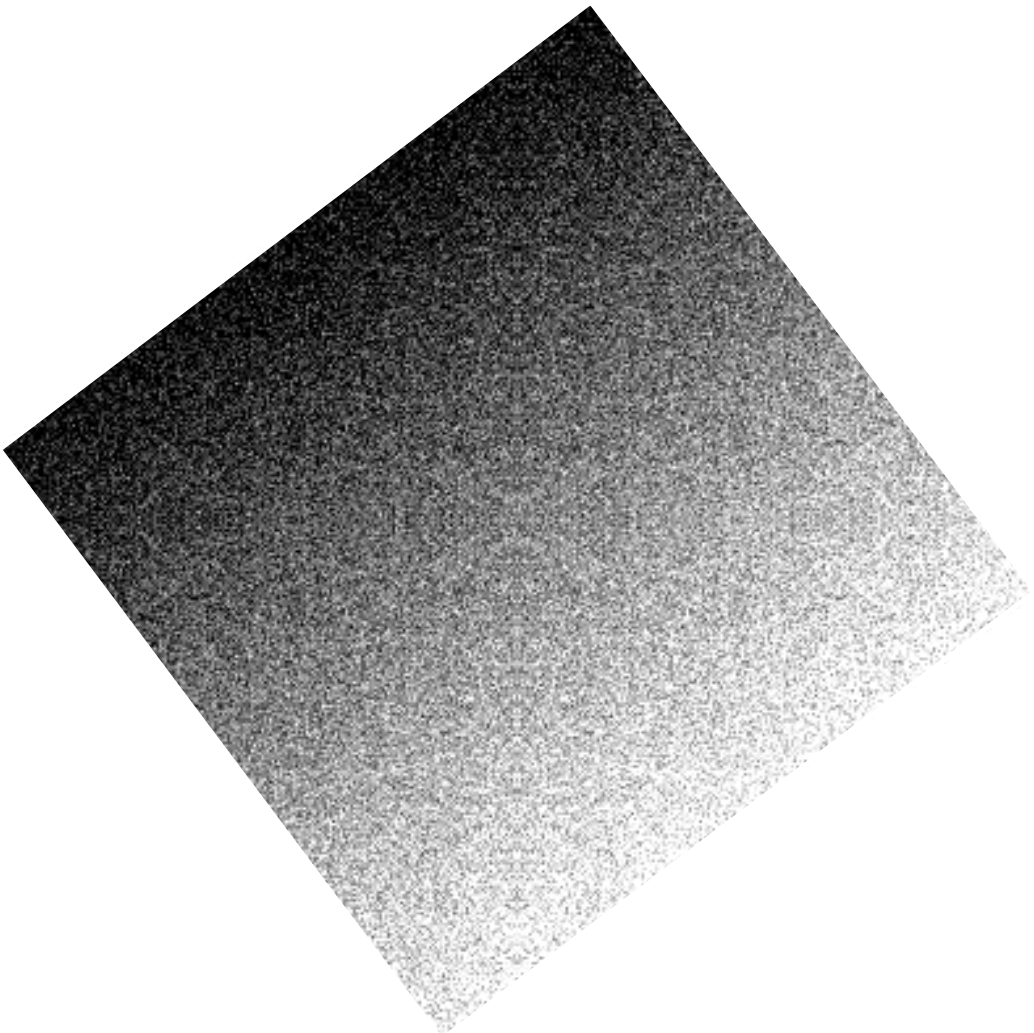
Intelligent Decision Server

IBM

Basic Tool User's Guide

Version 1



SC26-8908-00

Intelligent Decision Server

IBM

Basic Tool User's Guide

Version 1

SC26-8908-00

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page v.

First Edition (October 1996)

This edition applies to Version 1 of Intelligent Decision Server, 5639–IDS, and to any subsequent releases until otherwise indicated in new editions or technical newsletters. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 1996. All rights reserved.**
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Notices v
Trademarks v

About This Book vii
Who Should Read This Book vii
We'd Like Your Comments vii

Chapter 1. What Is the BASIC Tool? 1

Chapter 2. Getting Started 2
Using the BASIC Tool 3
Logging Your Session 4
Variations Between IBM BASIC and Microsoft BASIC 5

Chapter 3. Using the BASIC Tool in Stand-Alone Mode 6
Understanding Stand-Alone Mode Conventions 6
IDS BASIC Commands 6
Entering Your Program and Program Values 7
Modifying a Program 8
Writing Program Results 8
Printing Program Results 9

Chapter 4. Running a BASIC Program from a Spreadsheet 10
BASIC Program Syntax for Spreadsheets 10
Spreadsheet Cell Tags and Data Types 10
Using the New-Date Spreadsheet Cell Tag 11
Working with Spreadsheet Input and Output Regions 11
Opening and Closing Spreadsheet Input and Output Regions 12
Reading and Writing Dimensions of Input and Output Regions 12
Reading and Writing Spreadsheet Cell Types and Values 13
Using the End-of-Row Tag when Reading and Writing Spreadsheet Cell Values 14
Sample Program Segments 14
Typing a Program Run Statement in a Spreadsheet 16
Running a BASIC Program from a Spreadsheet 16
Running a BASIC Program with the Same Name as a BASIC icon 17
Running a BASIC Program in a BASIC Icon that is Contained in a Folder . . 17
Running Your Program in a Spreadsheet 17
You Try It – Transpose Data Within a Spreadsheet 19
Recalculating Spreadsheet Values 23
Tracing Data Transmission Between Spreadsheet Regions 24
Trapping Run-time Errors 24

Chapter 5. Writing BASIC Programs for Transformers 25
Using Keywords and Special Characters for Transformers 25
Understanding Keyword Conventions for Transformers 26

© Copyright IBM Corp. 1996

REM $$ INLABELS 26
REM $$ OUTLABELS 26
REM $$ ARGLABELS 26
REM $$ ARGVALUES 27
Running Your Program with a Transformer 27
You Try It – Copy Data to a Spreadsheet and Print a Log File 28

Chapter 6. Troubleshooting BASIC Programs 34
Programs for Spreadsheets 34
Programs for Transformers 34
Programs for Spreadsheets or Transformers 34

Appendix A. IBM BASIC Operators and Keyword Syntax 37

Glossary 44

IDS Publications 45

Index 46

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

Licensees of this program who wish to have information about it for the purpose of enabling (1) the exchange of information between independently created programs and other programs (including this one) and (2) the mutual use of the information that has been exchanged, should contact:

IBM Corporation
555 Bailey Avenue, W92/H3
P.O. Box 49023
San Jose, CA 95161–9023

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks

The following terms are trademarks of IBM Corporation in the United States or other countries or both:

ACF/VTAM	DXT	PROFS
AIX	Enterprise System/9000	PS/2
Application System/400	ES/9000	QMF
APPN	IBM	RACF
AS/400	MVS/ESA	RISC System/6000
Capsule	MVS/XA	RS/6000
DATABASE 2	OfficeVision	S/390

© Copyright IBM Corp. 1996

DataJoiner	OfficeVision/VM	SQL/DS
DB2	Operating System/2	System/360
DB2 for OS/2	Operating System/400	System/370
DB2/6000	OS/2	Virtual Machine/Enterprise Systems Architecture
DFSORT	OS/400	VM/ESA
Distributed Relational Database Architecture	Personal System/2	VTAM
DRDA	Presentation Manager	Win-OS/2

C-bus is a trademark of Corollary, Inc.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

About This Book

The *BASIC Tool User's Guide* contains information about the components, features, and functions of the BASIC tool. The BASIC tool uses IBM BASIC (Beginner's All-purpose Symbolic Instruction Code), which is a subset of **Microsoft BASIC **(MS-BASIC); IBM BASIC lets you run BASIC programs alone in the IBM Intelligent Decision Server (IDS) or with certain IDS tools. Although this guide refers to several BASIC commands and functions, it does not describe the MS-BASIC language itself. Refer to an MS-BASIC reference for information about MS-BASIC.

Who Should Read This Book

The information and instructions contained in this book are written on the premise that you have:

- Experience programming in BASIC
- A working knowledge of the IDS Spreadsheet, Capsule, and Transformer tools

We'd Like Your Comments

Your feedback is important to us in providing you with the most accurate and high-quality information. If you have any comments about this book or any other IDS documentation, please visit our Website at:

http://www.dss.ibm.com

There you'll find the Feedback Connection page where you can enter comments and send them to us.

© Copyright IBM Corp. 1996

Chapter 1. What Is the BASIC Tool?

The IBM Intelligent Decision Server (IDS) BASIC tool gives you access to an independent BASIC environment within which you can create and run Microsoft Beginner's All-purpose Symbolic Instruction Code (MS-BASIC) programs. In the BASIC environment, you can perform operations that MS-BASIC supports, such as load, run, and save programs and data files. The IDS BASIC tool uses IBM BASIC–a subset of MS-BASIC that allows the tool to run BASIC programs in IDS or to interact with the Spreadsheet tool or with an IDS transformer in a capsule tool.

You can create and store BASIC programs in the BASIC environment and run them three ways:

- In stand-alone mode, where the BASIC tool does not interact with other IDS tools. Chapter 3, “Using the BASIC Tool in Stand-Alone Mode” on page 6 describes the independent BASIC environment.
- From a Spreadsheet tool when the spreadsheet and the BASIC tool icon, containing the program that you wish to run, are at the same level. That is, both of the icons are on the desktop, in the same folder, or in the same capsule tool. A BASIC program can read cell values from one or more spreadsheet input regions, process the data, and write results to one or more spreadsheet output regions. Chapter 4, “Running a BASIC Program from a Spreadsheet” on page 10 explains how to run a BASIC program from a spreadsheet.
- From a Transformer tool in a capsule tool when both the transformer and the BASIC tool icon containing the program that you wish to run are in the capsule tool. Chapter 5, “Writing BASIC Programs for Transformers” on page 25 explains how to write a BASIC program that is compatible with Transformer tool format and how to run the program with the Transformer.

This book is not intended to explain the MS-BASIC programming language. Refer to an MS-BASIC language reference for information about MS-BASIC interpreter commands and syntax.

© Copyright IBM Corp. 1996

Chapter 2. Getting Started

The BASIC tool consists of one window that supports a BASIC environment. You can open a BASICLog window to log activities that occur in the BASIC tool window. This chapter describes the BASICLog windows and provides an overview of how to use the BASIC tool.

You can copy a BASIC tool icon to your desktop from the *Blank Icons file drawer within the New Icons file drawer.

The BASIC tool window contains a program area, as shown in Figure 1, in which you load, type, or copy your program and issue commands to the BASIC interpreter. Window buttons let you control window operations.

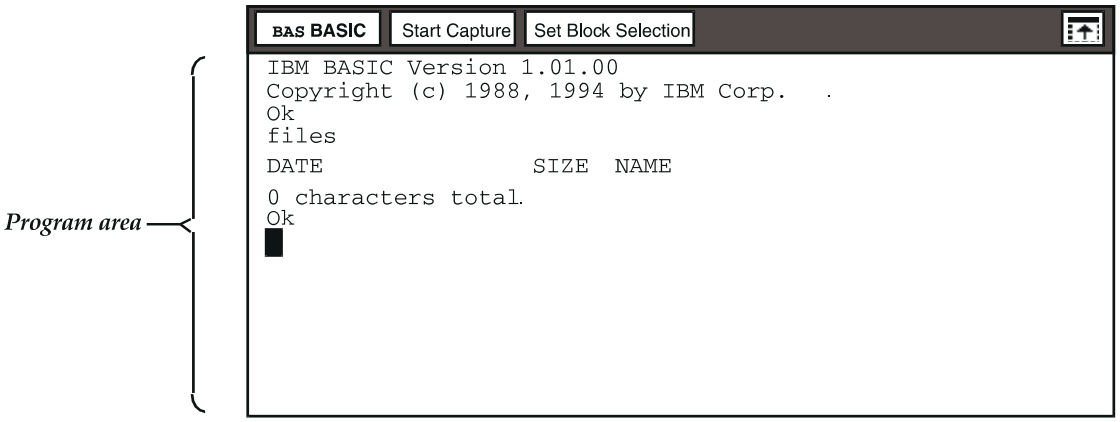


Figure 1. BASIC tool window

Table 1 lists the **BASIC** tool window buttons and describes their functions.

Table 1 (Page 1 of 2). BASIC tool window buttons

Button	Description
Start Capture	A button that opens a BASICLog window on the desktop.

© Copyright IBM Corp. 1996

Table 1 (Page 2 of 2). BASIC tool window buttons

Button	Description
Set Block Selection or **Set Line Selection**	A toggle button that controls the way BASIC window entries are selected with the mouse. When **Set Line Selection** is on, BASIC window entries are selected by sequential lines from the point of origin. When **Set Block Selection** is on, BASIC window entries are selected in a block from the point of origin. **Set Block Selection** lets you be more selective about the text you choose to copy from the BASIC tool window. When **Set Block Selection** appears in the window header, it means that line selection is activated. Conversely, when **Set Line Selection** appears in the window header, block selection is activated.

In the program area of the **BASIC** tool window, you can:

Type the BASIC statements for your program

Enter commands to the BASIC interpreter

View both general and error messages from the BASIC interpreter

The line in the BASIC tool window containing the word files and the header below it that displays DATE, SIZE, and NAME indicate that you can type FILES to list the programs that exist in that particular BASIC tool icon, the date of the last change, and the program size.

To activate the BASIC tool window, you must first deselect any objects outside the BASIC tool window.

Using the BASIC Tool

This section provides an overview of how to use the BASIC tool and capture program results. For specific information about using the tool:

in stand-alone mode, see Chapter 3, “Using the BASIC Tool in Stand-Alone Mode” on page 6.

with the Spreadsheet tool, see Chapter 4, “Running a BASIC Program from a Spreadsheet” on page 10.

with Transformer tools, see Chapter 5, “Writing BASIC Programs for Transformers” on page 25.

To use the BASIC tool:

1. Copy a BASIC tool icon to your desktop.
2. Verify in the Options window that the icon is not configured with read-only access. Read-only access does not enable you write to the icon.

3. Open the icon, and enter a new program into the program area of the BASIC tool window line by line.

   Copy a BASIC program from another BASIC tool window.

   You can also load an existing program into the working buffer by typing:

   LOAD "program_name"

   Methods of entering a program into the BASIC tool icon are also described in "Entering Your Program and Program Values" on page 7.

4. To save a new program, type:

   SAVE "program_name"

   Be sure to include quotes when using the SAVE command.

5. To record the program results, open a BASICLog window.

6. To run a program, type:

   RUN "program_name"

   If you want to run the current loaded program, you do not have to specify the program name; you can type RUN without specifying a program name.

7. You cannot save to a BASIC tool icon that has read-only access. If you enter a program and then realize that you do not have access privileges, to save it:

   open a BASICLog window

   in the BASIC tool window, display the program by typing LIST

   copy the listed program in blocks from the BASICLog window into another BASIC tool icon that has read-write access.

Logging Your Session

Use the BASICLog window to record your BASIC tool session.

To open a BASICLog window, click on the **Start Capture** button in the **BASIC** tool window header. A BASICLog window opens on the desktop.

A Text window captures anything that appears in the program area of the Text window. The BASICLog window also provides editing functions. For example, text within it can be scrolled, searched, and replaced. The BASICLog window is updated whenever you close it, or for each screen-full of information that it captures.

To close a BASICLog window, click on the title block in the BASICLog window header.

When a BASICLog window is closed, the date and time is included in the icon name, indicating when the window was opened.

To activate the BASICLog window, you must first deselect any objects outside the BASIC tool window.

Variations Between IBM BASIC and Microsoft BASIC

The functionality of the following keywords in IBM BASIC deviates from their usage in Microsoft BASIC (MS-BASIC).

Table 2. IBM BASIC keywords that deviate from MS-BASIC

Keyword	Function
CVL	A 4-byte string that casts a string value to a long integer value.
MKL	A 4-byte integer expression that casts a long integer value to a string value.
FILES	Lists all BASIC programs in the current working buffer of the BASIC tool, takes no argument, and is used only interactively.
FRE	Returns a constant greater than zero.
OPEN	Supports the "TOOL: ." device for spreadsheet regions; all other devices are assumed to be program names.
RENUM	Provides an optional parameter that permits renumbering to stop at a specified row number.

The following commands and functions are not supported in IDS BASIC. There is also no direct line printer support. If you attempt to use these commands and functions in IDS BASIC, the message Syntax Error is displayed.

AUTO	LLIST	POKE
BSAVE	LPOS	USR
CALL	LPRINT	VARPTR
CLS	LPT1	WAIT
DEFSEG	PEEK	WIDTH
EDIT		

Machine language subroutines, features dependent on absolute memory locations, and all graphical functions have also been removed from IBM BASIC.

See Appendix A, "IBM BASIC Operators and Keyword Syntax" on page 37 for a list of available IBM BASIC operators and keywords.

Chapter 3. Using the BASIC Tool in Stand-Alone Mode

Stand-alone mode refers to an independent BASIC environment that is available when you open a BASIC tool icon. Each BASIC tool icon provides an environment that is separate from IDS. The BASIC tool lets you load, run, save, and delete a BASIC program in stand-alone mode without any interaction with other IDS icons.

To illustrate how the BASIC tool environment is distinct from the IDS environment, when you use the KILL command to delete a program in a BASIC tool icon, a temporary copy of the program is not retained in the Discard icon. By contrast, in the IDS environment when you delete an icon and its contents, a temporary copy of the icon remains in the Discard icon.

You can have several BASIC tool icons on your desktop or in a folder. Each icon is its own stand-alone environment with its own file directory and working buffer. When you open a BASIC tool icon, a list of programs and data files is displayed for that particular stand-alone environment. Each BASIC tool icon can contain as many BASIC programs or data files as you want, but you can load or run only one program at a time. You can run several programs simultaneously from several different BASIC tool windows.

Understanding Stand-Alone Mode Conventions

The following conventions apply to the use of the BASIC tool in stand-alone mode:

The BASIC tool supports random-access and sequential data files. These files must reside in the same BASIC tool icon as the BASIC program that is running them.

Disk drive names must not be used as prefixes in program names. For example:

Instead of:	a:mystuff
Use:	mystuff

As with standard BASIC, the line-continuation character is a back slash (\). The back slash lets you enter lines longer than the length of a screen line.

IDS BASIC programs do not take a program name extension. Although the .BAS extension is conventional for MS-BASIC programs, you should not append .BAS to your IBM BASIC programs because, in IDS BASIC, a period is used to separate a BASIC tool icon name from a program name.

If you import a MS-BASIC program into the IDS environment, you must remove the .BAS extension from the program name.

IDS BASIC Commands

Table 3 on page 7 describes the MS-BASIC file-control commands that IBM BASIC uses. Refer to an MS-BASIC language reference for descriptions of BASIC interpreter commands and syntax.

© Copyright IBM Corp. 1996

Table 3. File-control commands supported by IBM BASIC

These commands or key combinations	Perform these functions
FILES	Lists the name, size, and the last-changed date of programs that exist in the current directory of a specific BASIC tool icon.
KILL *program_name*	Deletes the specified program from the current directory. When you use the KILL command to delete a BASIC program, a temporary copy of the program is not retained in the Discard folder on your desktop.
LIST or **LIST 10-50**	Displays the entire contents or a program segment of the currently loaded program.
LOAD *program_name*	Loads the specified program into the working buffer.
NAME *old_program_name* **AS** *new_program_name*	Renames the specified program.
NEW	Clears the current program from the working buffer.
RUN	Runs the current program.
Ctrl + [C]	Stops the current program.
CONTINUE "program_name" RUN	Restarts the specified program at the point it was interrupted.
SAVE *program_name*	Saves a copy of the specified program.

If you want to list, run, or save a program that is currently in the working buffer of a BASIC tool icon, it is not necessary to specify a program name when using the LIST, or RUN commands.

Entering Your Program and Program Values

Each BASIC tool icon provides an independent BASIC environment. This means that for data, programs, or subroutines to work together with a program, they must all reside in the same BASIC tool icon.

You can enter a program in the BASIC tool icon by:

- Typing your program, line by line, into the program area of the BASIC tool window.
- Loading an existing program into the BASIC tool's working buffer using the LOAD command.
- Listing the program in the program area of the BASIC tool window using the LIST command and copying the program, block by block if it is long, into the program area of another BASIC tool window.

You can enter values to a BASIC program:

Interactively, by entering values for program variables using INPUT statements in the BASIC program.

Non-interactively, by reading values into program variables using READ and DATA statements in the BASIC program.

Modifying a Program

You can modify a BASIC program two ways. The first way is to work with a program segment:

1. In the program area of the BASIC tool window where the program to be modified resides, list program segments using the LIST command. For example:

   ```
   LIST 1 -5
   ```

2. Retype the lines that you want to modify, using the listed lines as a reference if necessary.

   As you enter each line, the modification replaces the original code.

3. Save the program using the SAVE commands. Remember to include quotes around the program name.

The second way is to list the program in a BASICLog window:

1. Open a BASICLog window.
2. In the program area of a BASIC tool window, list the program using the BASICLog window.
3. Modify the program in the BASICLog window, where you can scroll through the program.
4. Delete the old program in the BASIC tool icon using the KILL command.
5. When you are finished modifying the program in the BASICLog window, copy it in segments back to the BASIC tool.
6. Save the program using the SAVE command. Remember to include quotes around the program name.

Writing Program Results

BASIC program results can be written to:

The program area of the BASIC tool window in which you are working. This is the default and is done using PRINT statements in the program.

Another program within the BASIC tool icon where you are working. Within your program, include statements to open another program and write to it.

A BASICLog window.

A spreadsheet region. See Chapter 4, “Running a BASIC Program from a Spreadsheet” on page 10 for details.

Another file using a capsule tool. See Chapter 5, “Writing BASIC Programs for Transformers” on page 25 for details.

Printing Program Results

To print the results of your BASIC program:

1. Capture what you want to print in a BASICLog window.
2. Edit or format the captured program output.
3. Close the BASICLog window.
4. Copy the icon to any printer icon on your desktop.

Chapter 4. Running a BASIC Program from a Spreadsheet

BASIC programs written in the stand-alone environment can be run from a spreadsheet. Such programs can read the cell values from one or more input regions of the spreadsheet, process data, and write the results as cell values to one or more output regions. BASIC reads and writes spreadsheet regions by row, and handles all the cell data types produced by a spreadsheet (integer, real, string, Boolean, date, and N/A).

Input and output regions are named portions of a spreadsheet. A BASIC program reads data from input regions and writes data to output regions. For more information about spreadsheets, see *Spreadsheet User's Guide* .

Steps and requirements for running a BASIC program from a spreadsheet are explained in “Running Your Program in a Spreadsheet” on page 17.

BASIC Program Syntax for Spreadsheets

The following sections describe spreadsheet cell tags and corresponding data types and show how to work with spreadsheet input and output regions.

Spreadsheet Cell Tags and Data Types

Spreadsheet cell tags are used in BASIC programs to read data from and write data to spreadsheet regions. A BASIC program reads from and writes to spreadsheet regions as follows:

A WRITE statement writes data to a spreadsheet output region and requires a spreadsheet cell tag that corresponds to a spreadsheet data type.

When writing to a spreadsheet output region, for each spreadsheet row specified in your program, you must include an end-of-row tag (–3).

A READ statement reads data from a spreadsheet input region; the spreadsheet cell tag that identifies a data type is automatically attached to data that is read.

Table 4 lists spreadsheet cell tags and their corresponding spreadsheet data types. The tags, do not take spreadsheet cell values and cause program errors if you include a value when using them. Examples of how spreadsheet cell tags are used in a BASIC program are shown in Figure 2 on page 15 and Figure 3 on page 16.

Table 4 (Page 1 of 2). Spreadsheet cell tags

Tag	Data type	Not a data type
0	Integer	
1	Real	
2	String	
4	Boolean	

© Copyright IBM Corp. 1996

Table 4 (Page 2 of 2). Spreadsheet cell tags

Tag	Data type	Not a data type
9	New date	
–3		End-of-row
–1		N/A (null)
–2		Error
5		Unspecified value (for example, empty)

Using the New-Date Spreadsheet Cell Tag

Programs that use the old date type must be rewritten to handle the new-date type.

The new-date type must be written in the form 9**,month,year,resolution,** as follows:

9,January, 1994,3

where 9 is the spreadsheet cell tag for new-date types, and 3 (month) is the date resolution.

Table 5 lists the date resolution codes.

Table 5. Date resolution codes for the new-date cell tag

Resolution	Description
0	Day
1	Week
2	4-week period
3	Month
4	Odd bimonth, where the calendar year is divided into 2-month periods ending in an odd-numbered month: FM, AM, JJ, AS, ON, DJ
5	Quarter
6	Year
7	Even bimonth, where the calendar year is divided into 2-month periods ending in an even-numbered month: JF, MA, MJ, JA, SO, ND

Working with Spreadsheet Input and Output Regions

The following sections provide descriptions of how to open and close spreadsheet input and output regions, how to read spreadsheet cell dimensions, and how to use the end-of-row tag (–3) when reading and writing values in spreadsheets. Program segments are provided to illustrate the program syntax for IBM BASIC.

Opening and Closing Spreadsheet Input and Output Regions

To open a spreadsheet region for input or output, include an OPEN statement in your program, using this format:

For an input region:	OPEN “TOOL:BASICIN” FOR INPUT AS #5
For an output region:	OPEN “TOOL:BASICOUT” FOR OUTPUT AS #5

where #5 is a variable identifier that you assign to the current input or output region, and TOOL:BASICIN and TOOL:BASICOUT are always uppercase.

The **#** symbol is optional in OPEN statements. The identifier must be a numeric, such as 5, shown in the preceding examples.

The TOOL statement is used to interact with a spreadsheet. Because a BASIC program does not recognize the names of the spreadsheet input and output regions, it uses these standard names:

BASICIN	reads input from a spreadsheet input region.
BASICOUT	writes output to a spreadsheet output region.

TOOL:BASICIN and TOOL:BASICOUT are always uppercase.

For a program that reads multiple input regions or writes multiple output regions successively, each BASICIN and BASICOUT statement opens the next specified region that appears in the list of input or output regions in the BASIC program call. If the program attempts to open more input or output regions than exist in a spreadsheet, the error message, Permission Denied, appears in the BASIC tool window.

Only one input region and one output region can be open at the same time. Other OPEN and CLOSE statements (such as those used for random-access disk files) can be nested within or around TOOL statements used to open input and output regions.

An open input or output region must be closed before the next successive input or output region can be opened. To close a region, include a CLOSE statement in your program, using the format:

```
CLOSE #5
```

where #5 is the identifier currently being used for corresponding input and output regions.

Reading and Writing Dimensions of Input and Output Regions

The first two values to be read from an input region or written to an output region must be the same dimension (in rows and columns) as the input or output region itself.

To read the dimensions of an input region, include an INPUT statement after the OPEN statement, as follows:

```
OPEN “TOOL:BASICIN” FOR INPUT AS #5
```

```
INPUT #5, NROWS, NCOLS
```

where:

#5	is the variable identifier that you assigned in the TOOL statement to open the input region.
NROWS	is the number of rows read from the current spreadsheet input region.
NCOLS	is the number of columns read from the current spreadsheet input region.

If input data is a string value, the string cannot exceed 253 characters, or it will be truncated, and a message will appear in the Important Message window.

To write the dimensions of an output region, include a WRITE statement after the OPEN statement, as follows:

```
OPEN "TOOL:BASICOUT" FOR OUTPUT AS #5
WRITE #5, NROWS, NCOLS
```

where:

#5	is the same identifier that you used to open the corresponding input region.
NROWS	is the number of rows written to the current spreadsheet input region.
NCOLS	is the number of columns written to the current spreadsheet input region.

If the value of either dimension is negative or zero, the error message Device I/O appears in the BASIC tool window.

Reading and Writing Spreadsheet Cell Types and Values

After the statement that reads or writes the dimensions of spreadsheet regions, include a statement to read or write spreadsheet cell types and values to the regions. Be sure to precede a cell value with the cell type, as shown in the following example. The cell type and cell value must be written with the same WRITE statement. For example:

```
IF TYPE = 2 THEN WRITE #5, TYPE, VALUE
```

where:

TYPE	is a value whose data type the IBM BASIC interpreter checks; if the condition tests true, the value is written to a spreadsheet output region. In this case, if TYPE is a string, the string is written to output region #5.
2	defines a spreadsheet string type.
#5	is the variable identifier that you assigned to the current output region and that also corresponds to the identifier for the current input region.
VALUE	is a string that you provide.

A spreadsheet cell tag and its cell value can be read with individual INPUT statements:

```
INPUT #5, TYPE
```

```
INPUT #5, VALUE
```

with the same INPUT statement:

```
INPUT #5, TYPE, VALUE
```

The cell tags and cell values of different cells cannot be mixed in the same INPUT statement. Also, cell tags and cell values must be written with the same WRITE statement.

The cell tags for unspecified values (**5**), NA (**–1**), error (**–2**), and end-of-row (**–3**) must also be written when appropriate, but they should not be followed by values.

Include an end-of-row tag at the end of each spreadsheet row with a separate WRITE statement. For example:

```
WRITE #5, ENDOFROW
```

Using the End-of-Row Tag when Reading and Writing Spreadsheet Cell Values

The use of the end-of-row cell tag (–3) is slightly different for reading and writing values. When reading values from a spreadsheet, the format is:

```
INPUT#5, TYPE
```

where TYPE is a data value that the IBM BASIC interpreter checks and writes to a spreadsheet output region if a condition tests true.

In a FORloop, such as the one shown in Figure 2 on page 15, values are read into input region #5 as data types (line 180). A value of –3 read into the input region indicates that the end of the spreadsheet row has been reached (line 230). The NEXT ROW statement then advances the program input to the next spreadsheet row (line 240).

When writing values from a spreadsheet, the format for indicating the end of a row is:

```
WRITE #5, ENDOFROW
```

A standard value of –3 is assigned to ENDOFROW (line 130 in the variable assignments shown in Figure 2 on page 15) to indicate the end of the spreadsheet row.

Sample Program Segments

Figure 2 on page 15 lists a program segment containing statements to open, close, and read the values of an input region.

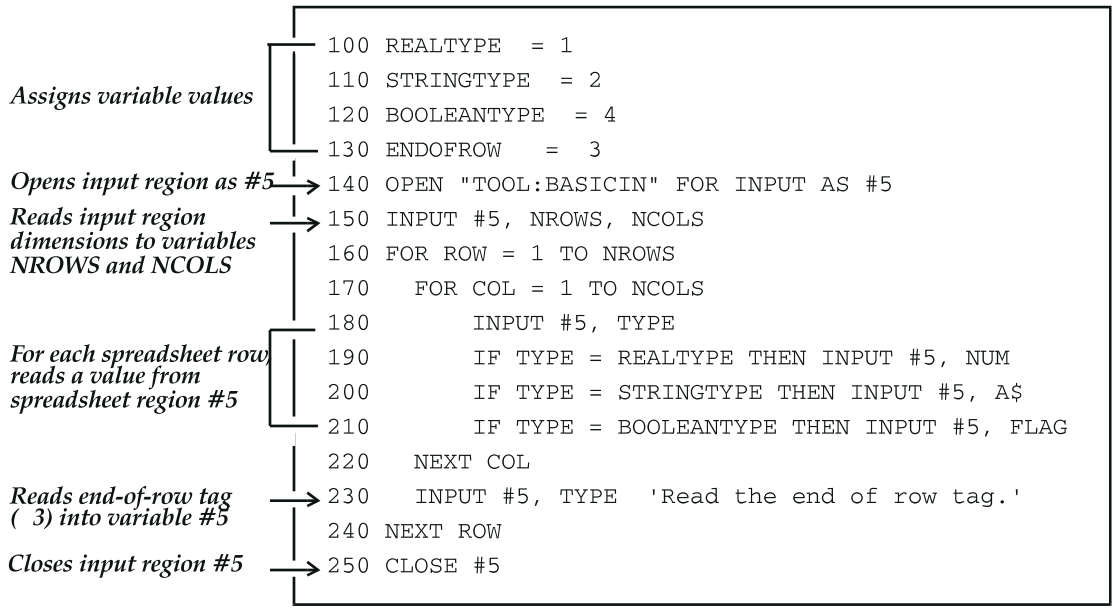


Figure 2. Syntax for working with a spreadsheet input region

A BASIC program can easily get out of synchronization when reading spreadsheet cell tags and cell values. Remember that some cell tags should not be followed by cell values. For a list of such cell tags, see Table 4 on page 10.

Figure 3 on page 16 lists a program segment containing statements to open, close, and write the values of an output region.

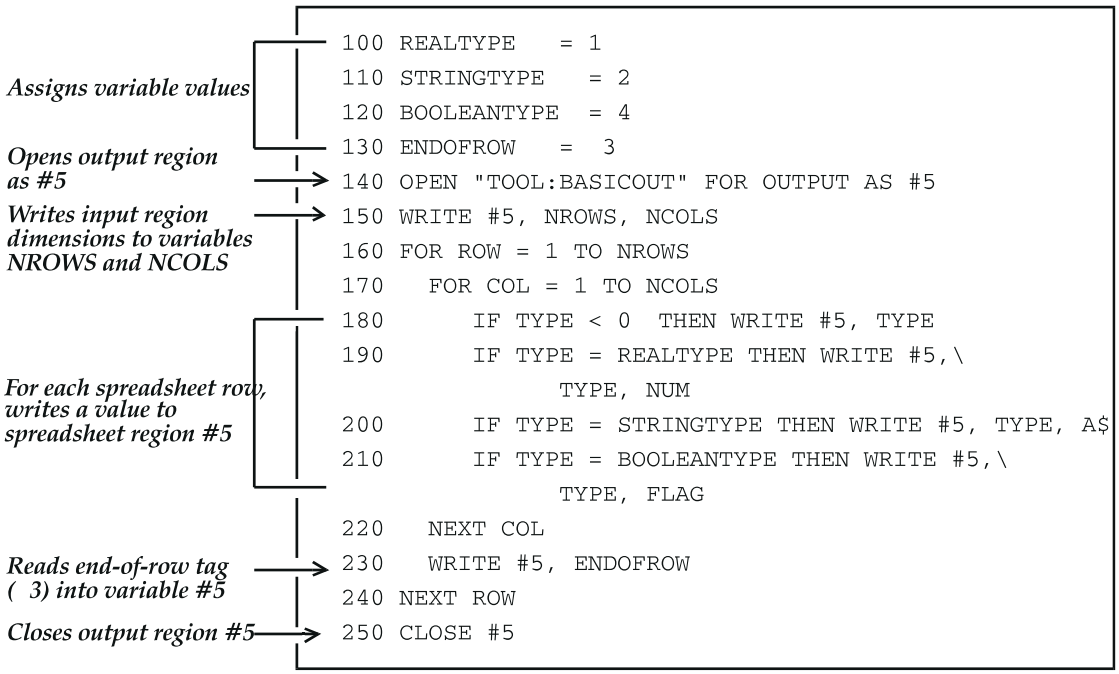


Figure 3. Syntax for working with a spreadsheet output region

Typing a Program Run Statement in a Spreadsheet

The following sections describe various ways to type the program run statement depending on the location of the BASIC tool icon. If a program is not found in the BASIC tool icon that you specify, BasicLog icon with a date and time stamp) appears on the desktop with the same name as the BASIC program. When you open the BasicLog icon, the following message displays in the window:

program_name not found

Running a BASIC Program from a Spreadsheet

1. To run a program from an IDS Spreadsheet tool, type the following in a spreadsheet cell:

BASIC Met Stat.Multivariate Regression([Sample Data];[Results])

where:

Met Stat	is the name of the BASIC icon that contains the Multivariate Regression program.
Multivariate Regression	is the name of the BASIC program.
Sample Data	is the input region.

Results	is the output region.

Input and output regions must be enclosed in brackets and separated by a semicolon.

Running a BASIC Program with the Same Name as a BASIC icon

1. To run a program with the same name as a BASIC icon, type only one name, as follows:

```
BASIC Transpose ([IN];[OUT])
```

where:

Transpose	is the name of the BASIC program.
IN	is the input region.
OUT	is the output region.

where IN and OUT are the input and output regions.

Running a BASIC Program in a BASIC Icon that is Contained in a Folder

1. To run a program that is in a BASIC icon inside a folder type:

```
BASIC Math Library|Transpose ([IN];[OUT])
```

where:

Transpose	is the name of the BASIC icon.
Math Library	is the name of the folder that contains the BASIC icon.

Path name components are separated by a pipe (|).

Avoid using the same icon name for several BASIC icons. If you have multiple icons with the same name, only one is checked for the specified BASIC program, which could result in the BASIC interpreter locating and running the wrong program.

Running Your Program in a Spreadsheet

After you have entered a program run statement in a spreadsheet, you can run the program by selecting Run BASIC Program from the Spreadsheet **Special** menu.

You can also run a BASIC program from a spreadsheet in a capsule tool if you want to write spreadsheet results to another icon, such as another spreadsheet.

To run a BASIC program from a spreadsheet in a capsule tool, as shown in Figure 4 on page 18, you must include at least the following items in the capsule window:

a Spreadsheet icon	This contains the program run statement. The Spreadsheet icon in the example on Figure 5 on page 19 is named Input.

a BASIC tool icon	This is required because it contains the program that is executed by a run statement in the spreadsheet. The BASIC tool icon in the example on Figure 5 on page 19 is named Matrix.

When running a BASIC program from a spreadsheet, the BASIC tool icon and the Spreadsheet icon that contains the program run statement must be located at the same level in IDS. For example, they must both be in the same folder, within the same capsule tool, or on your desktop. In the example on Figure 5 on page 19, the Spreadsheet and BASIC tool icons are in the same capsule tool.

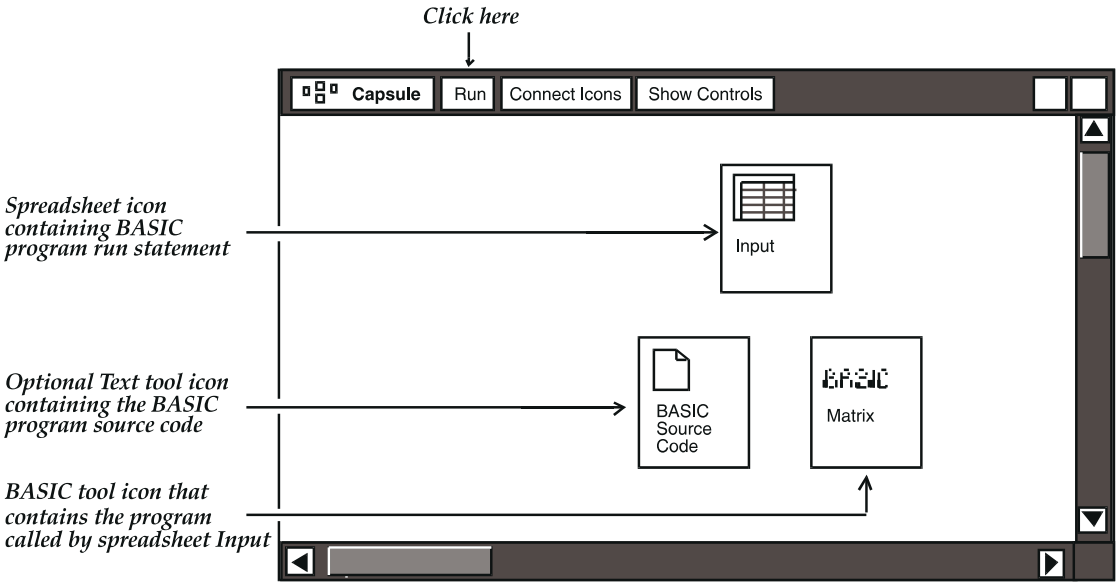


Figure 4. Running a BASIC program in a capsule tool

To run the program, click on the **Run** button in the capsule window header.

Typically, you would run a program in a capsule tool either:

As a convenience to contain your program source code, a BASIC icon, and an input icon–such as a spreadsheet–at the same level, as shown in the preceding example.

If you want to write spreadsheet results to another icon, such as another spreadsheet, you can include another icon in the capsule tool to read the program results. Figure 5 on page 19 shows an example of this setup.

Although the capsule tool does not have to include a Text icon that contains the BASIC program source code, it is convenient because it lets you view the program and modify it as needed.

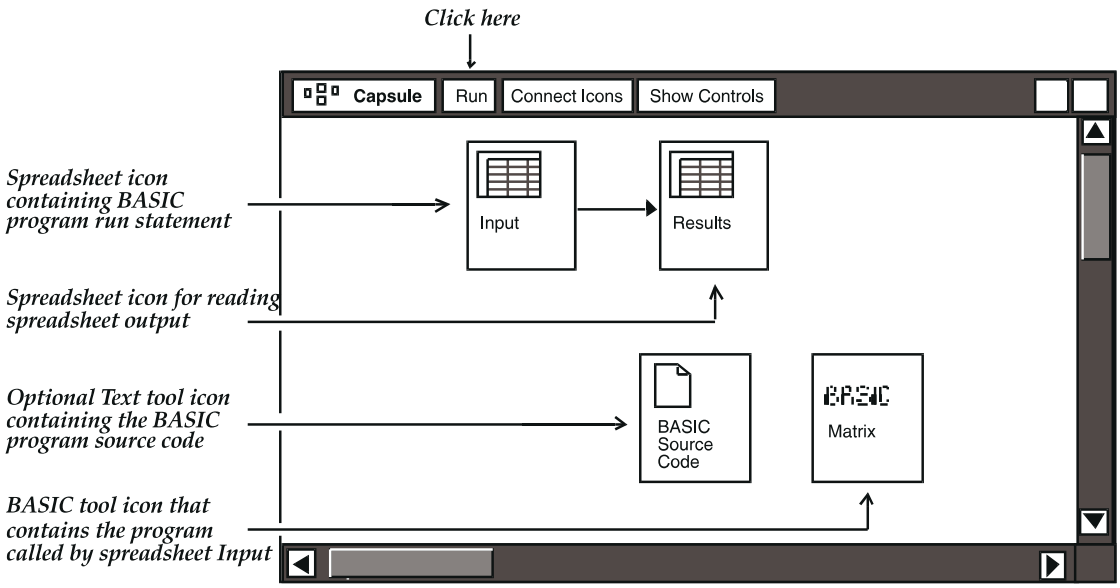


Figure 5. Writing spreadsheet results to another icon using a capsule tool

To run the program, click on the **Run** button in the capsule window header.

You Try It – Transpose Data Within a Spreadsheet

The following sample BASIC program reads two spreadsheet regions of the same dimension, as shown in Figure 6 on page 22. It then sums the two input regions to produce the first output region, and it transposes two cell values in the first input region to produce the second output region. All cell values are assumed to be of type real. To run this sample program:

1. Open a BASIC tool icon, verify that it does not have read-only access, and enter the following program in it:

```
1  OPTION BASE 1
11  REALTYPE = 1
14  ENDOFROW = -3
15  REM
16  REM     Input Region 1
17  REM
18  OPEN "TOOL:BASICIN" FOR INPUT AS #5
19  INPUT #5, NROWS1, NCOLS1
2  NROWS3 = NROWS1    :    NCOLS3 = NCOLS1
21  NROWS4 = NROWS1    :    NCOLS4 = NCOLS1
22  DIM INT1(NROWS1, NCOLS1), OUT3(NROWS3, NROWS4),    \
```

```
 OUT4(NROWS4,  NCOLS4)
23  FOR ROW = 1 TO NROWS1
24        FOR COL = 1 TO NCOLS1
25                INPUT #5, TYPE, INT1(ROW,COL)
26     NEXT  COL
27      INPUT #5, TYPE              ' Have to read end of row
28  NEXT ROW
29  CLOSE #5
3   REM
31  REM     Input Region 2
32  REM
33  OPEN “TOOL:BASICIN” FOR INPUT AS #5
34  INPUT #5, NROWS2, NCOLS2
345 DIM IN2(NROWS1, NCOLS1)
35  FOR ROW = 1 TO NROWS2
36      FOR COL = 1 TO NCOLS2
37                INPUT #5, TYPE, INT2(ROW,COL)
38     NEXT  COL
39      INPUT #5, TYPE              ' Have to read end of row
4   NEXT ROW
41  CLOSE #5
42  REM
43  REM     Region 3 = Region 1 + Region 2
44  REM
45  FOR ROW = 1 TO NROWS3
46      FOR COL = 1 TO NCOLS3
47                OUT3(ROW, COL) = INT1(ROW,COL) + INT2(ROW,COL)
48     NEXT  COL
49  NEXT ROW
5   REM
51  REM     Region 4 = Transpose of Region 1
52  REM
53  FOR ROW = 1 TO NROWS4
54      FOR COL = 1 TO NCOLS4
55                OUT4(ROW, COL) = INT1(COL, ROW)
```

```
56     NEXT  COL
57  NEXT ROW
58  REM
59  REM     Output Region 3
6   REM
61  OPEN "TOOL:BASICOUT" FOR OUTPUT AS #7
62  WRITE #7, NROWS3, NCOLS3
63  FOR ROW = 1 TO NROWS3
64      FOR COL = 1 TO NCOLS3
65              WRITE #7, REALTYPE, OUT3(ROW, COL)
66     NEXT  COL
67      WRITE #7, ENDOFROW
68  NEXT ROW
69  CLOSE #7
7   REM
71  REM     Output Region 4
72  REM
73  OPEN "TOOL:BASICOUT" FOR OUTPUT AS #7
74  WRITE #7, NROWS4, NCOLS4
75  FOR ROW = 1 TO NROWS4
76      FOR COL = 1 TO NCOLS4
77              WRITE #7, REALTYPE, OUT4(ROW, COL)
78     NEXT  COL
79      WRITE #7, ENDOFROW
8   NEXT ROW
81  CLOSE #7
82  REM
83  END
```

2. Save the program. In the program area of the BASIC tool window, type:

   ```
   SAVE "program_name"
   ```

   Be sure to include quotes when using the SAVE command.

3. Close the BASIC icon and name it.
4. Copy a Spreadsheet icon to your desktop and name it.
5. Open the Spreadsheet icon and, using the spreadsheet setup in Figure 6 on page 22, create two input regions and two output regions, and enter data in the input regions.

	A	B	C	D	E
1	Input 1			Input 2	
2	3.00	5.00		2.00	8.00
3	7.00	2.00		9.00	7.00
4					
5		BASIC Matrix ([Input 1], [Input 2]; [Output 1], [Output 2])			
6					
7	Output 1			Output 2	
8					
9					
10					

Figure 6. Spreadsheet input for sample program

6. Select a spreadsheet cell and enter a run statement directly into it, as shown in Figure 6. Alternatively, you can use the Spreadsheet Options window for a cell:
   a. Select a cell and open the Spreadsheet Options window.
   b. In the Spreadsheet Options window, select **Entry**.
   c. In the **Type** field, select **Text**.
   d. Type the program run statement in the **Entry** field that displays.

To enter a program run statement, type BASIC followed by the icon path name, and specify the input and output regions as follows. You can specify a single input and output region, as shown in this example:

```
BASIC icon_name.program_name ([Input]; [Output])
```

or you can specify multiple regions as follows:

```
BASIC icon_name.program_name ([Input1], [Input2]; [Output1], [Output2])
```

where:

icon_name.program_name	is the name of the BASIC icon and the BASIC program contained in the icon. A period is used to separate the icon name from the program name. If the program name and icon name are identical, you only need to specify one name. For more information on path name format, see "Running a BASIC Program in a BASIC Icon that is Contained in a Folder" on page 17.
[Input] or [Input1], [Input2]	is the name of a single input region, or the names of multiple input regions in order of use. With multiple regions, each name must be enclosed in brackets [ ] and separated from the next by a comma.

;	separates input and output region lists; if there is no region, a semi-colon must still be used. For example: BASICicon_name.program_name([Input];)
[Output] or [Output1], [Output2]	is the name of a single output region, or the names of multiple output regions in order of use. With multiple regions, each name must be enclosed in brackets [ ] and separated from the next by a comma.

The spreadsheet input and output regions for the sample program are labeled Input 1, Input 2, Output 1, Output 2.

1. Select the spreadsheet cell that contains the BASIC program run statement.
2. Select **Run BASIC Program** from the spreadsheet **Special** menu. The results should look like the spreadsheet shown in Figure 7. The output is read to regions Output 1 and Output 2.

   Numeric output is already set in the spreadsheet to Number.

You can stop a program at any time by pressing Ctrl + [C].

A BASIC program uses spreadsheet regions one at a time, working from left to right through the argument list in the program run statement. It can change the size of an output region (for example, when writing to a region), or skip rows and columns of a region when data is input from a spreadsheet (for example, when reading from a region).

	A	B	C	D	E
1	Input 1			Input 2	
2	3.00	5.00		2.00	8.00
3	7.00	2.00		9.00	7.00
4					
5		BASIC Matrix ([Input 1], [Input 2]; [Output 1], [Output 2])			
6					
7	Output 1			Output 2	
8	5.00	13.00		3.00	7.00
9	16.00	9.00		5.00	2.00
10					

Figure 7. Spreadsheet output from sample program

Recalculating Spreadsheet Values

Spreadsheet values must be recalculated after modifications by one or more BASIC programs.

1. To recalculate spreadsheet values after running a BASIC program, type /r at the end of the BASIC statement that you enter in the **Entry** field for a spreadsheet cell, as follows:

   BASIC Matrix ([Input1], [Input2]; [Output1], [Output2])/r

2. Click on Spreadsheet window header.

Tracing Data Transmission Between Spreadsheet Regions

When transmitting data between spreadsheet regions and a BASIC program, you can monitor the progress of transmission in a BASICLog window. If problems occur in data transmission, you can diagnose them by selectively using the trace option, TRON/TROFF ("trace on/off") in conjunction with the PRINT (to screen) statement.

To monitor data transmission to or from a particular region, append \t or \T, to the standard input and output names BASICIN or BASICOUT in the OPEN statement. For example:

```
OPEN "TOOL:BASICIN\t" FOR INPUT AS #1
```

This causes spreadsheet cell tags, data being read from the spreadsheet, and a BASIC program to be written to a BASICLog icon that appears on the desktop. A message also appears in the BASICLog icon, describing any transmission problems.

Trapping Run-time Errors

When a run-time error is trapped while running a BASIC program from a spreadsheet, an ON ERROR GOTO statement directs the program to an error handling routine. If an error occurs and there is no ON ERROR GOTO statement, a standard BASIC error message is sent to the BASICLog window and the program stops. Output to the spreadsheet might be incomplete, depending on where the BASIC run-time error occurred.

Chapter 5. Writing BASIC Programs for Transformers

The Transformer tool lets you work with tabular data, such as in a spreadsheet. You can write your own BASIC programs to control the prompts in the Transformer tool output region and to control how a transformer reads parameter values. Additionally, BASIC programs written for transformers read and process data from spreadsheet input regions and write data to spreadsheet output regions.

This chapter describes how to write BASIC programs that are compatible with Transformer tool format. Requirements for running a BASIC program with a Transformer are explained in “Running Your Program with a Transformer” on page 27.

For more information about using the Transformer tool, refer to the IDS *Transformers User's Guide*.

Using Keywords and Special Characters for Transformers

For BASIC programs that are written for transformers, special keywords must precede executable statements to specify certain values that are used in the transformer. These keywords, which can appear in any order in the program, are as follows:

INLABELS	specifies spreadsheet input region names.
OUTLABELS	specifies spreadsheet output region names.
ARGLABELS	specifies parameter descriptions and examples.
ARGVALUES	specifies default values for parameters.

The following special characters enable a BASIC program to work in conjunction with the Transformer tool:

REM $$	precedes lines that contain a keyword or prompt label.
>	indicates the start of a prompt label or a default value.

In the program segment shown in Figure 8 on page 26, lines 120, 140, 160, and 190 contain keywords. The lines that follow each keyword contain prompt labels or default values.

© Copyright IBM Corp. 1996

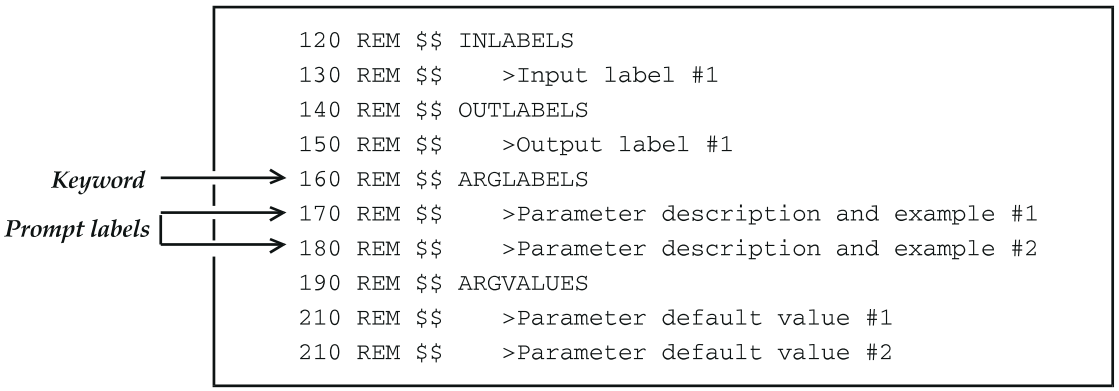


Figure 8. Syntax for working with transformers

Understanding Keyword Conventions for Transformers

The keyword conventions described in this section are recommended to make the format of your BASIC program compatible with transformers. It is conventional to begin the description with an initially capitalized word, as shown in the following examples.

REM $$ INLABELS

Use any name for the input region. For example:

```
REM $$ >Table_to_be_cleaned
```

REM $$ OUTLABELS

Use any name for the output region. For example:

```
REM $$ >Cleaned_table
```

REM $$ ARGLABELS

ARGLABELS contains a parameter description with an example of the values. Each ARGLABELS has a corresponding ARGVALUES. For example, the ARGLABELS parameter specified in line 170 is assigned the default value defined in line 200. Similarly, the ARGLABELS parameter and ARGVALUES default in lines 180 and 210 correspond to each other.

```
16  REM $$ ARGLABELS
17  REM $$          >Parameter description and example #1
18  REM $$          >Parameter description and example #2
19  REM $$ ARGVALUES
2   REM $$          >Parameter default value #1
21  REM $$          >Parameter default value #2
```

The default value is conventionally the first argument that appears in the argument list within parentheses. Typically, each example in the list is separated from the others by a semicolon and one or two spaces. For example:

```
REM $$  >Clean rows (Yes; No)
```

If all the valid arguments are given in the example list, capital letters can be used to indicate the uniqueness of the arguments. This allows you to type only the minimum initial characters needed to distinguish between values. For example:

```
(One; TWo; THree; FOur; FIve)
```

lets you type O for one, TW for two, TH for three, and so on.

If only a few of the possible values are given, an ellipsis (. . .) indicates that there are several more possibilities, as follows:

```
(all; 1; 1,2,3; )
```

The number of characters that you can enter when using ARGLABELS is limited to the line length of the BASIC tool window, which is 80 characters. You can exceed the maximum line length by using a back slash (\) for line continuations. The program listing on "You Try It – Copy Data to a Spreadsheet and Print a Log File" on page 28 shows an example of this.

REM $$ ARGVALUES

ARGVALUES is used to define a default value for an ARGLABELS parameter. The default is conventionally the first argument that appears in the argument list within parentheses. For example, for the following ARGLABELS:

```
REM $$  >Remove Integers: (Yes; No)
```

The corresponding default ARGVALUES is:

```
REM $$  >Yes
```

An example of the association between ARGLABELS and ARGVALUES is shown in lines 160 to 217 in the sample program on "You Try It – Copy Data to a Spreadsheet and Print a Log File" on page 28.

Running Your Program with a Transformer

To run a BASIC program with a Transformer, as shown in Figure 9 on page 28, you must include the following items in the capsule window:

two Spreadsheet icons	One Spreadsheet icon contains program input and the program run statement. In the example on Figure 9 on page 28, the Spreadsheet icon is named Input. Program results are written to the second Spreadsheet icon named Results.

a BASIC tool icon	This is required because it contains the program that is executed by a run statement in the spreadsheet named Input. The BASIC tool icon in the example on Figure 9 on page 28 is named Mydata.
a Transformer icon	This connects the Spreadsheet icons. The name of the BASIC program to be run is specified in the Transformer Controls window.

When running a BASIC program from a spreadsheet, the BASIC tool icon and the Spreadsheet icon that contains the program run statement must be located at the same level in IDS. For example, they must both be in the same folder, within the same capsule tool, or on your desktop. In the following example, the Spreadsheet and BASIC tool icons are in the same capsule tool.

In addition, you might also wish to include in the capsule tool a Text icon that contains the BASIC program source code so that you can view the program and modify it as needed. The Text tool in Figure 9 is named BASIC Source Code.

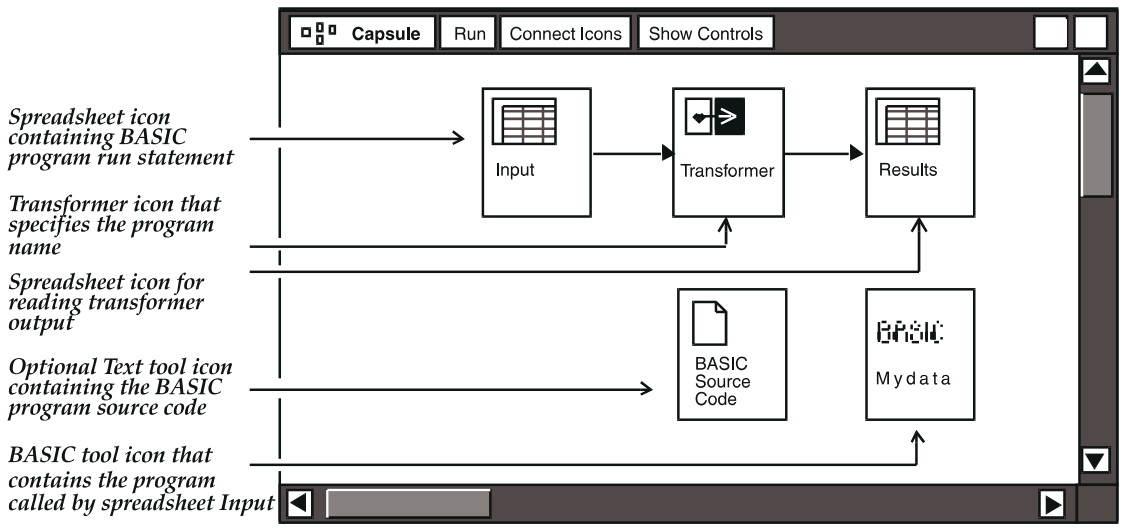


Figure 9. Running a BASIC program with a transformer in a capsule tool

You Try It – Copy Data to a Spreadsheet and Print a Log File

The following sample BASIC program shows how a transformer accepts input values. The program includes PRINT statements that format and direct output to a BASICLog icon. To run this sample program in IDS:

1. Open a BASIC icon and enter the following program in it:

```
1 REM        DEMONSTRATION PROGRAM FOR Transformer
1 1 REM
```

```
11  REM This section contains the keywords and their values
111 REM
12  REM $$ INLABELS
13  REM $$       >Input Demo #1
14  REM $$ OUTLABELS
15  REM $$       >Output Demo #1
16  REM $$ ARGLABELS
17  REM $$       >Remove Integers: (Yes;    No)
171 REM $$       >Remove Reals: (Yes;    No)
172 REM $$       >Remove Strings: (Yes;    No)
173 REM $$       >Remove Old Dates: (Yes;    No)
174 REM $$       >Remove Booleans: (Yes;    No)
177 REM $$       >Remove New Dates: (Yes;    No)
2   REM $$ ARGVALUES
21  REM $$       >No
211 REM $$       >No
212 REM $$       >No
213 REM $$       >No
214 REM $$       >No
217 REM $$       >No
218 REM $$       >
24  REM
25  REM This section reads one row of user entered parameter
26  REM values at a time and writes them into TOOL:BASICARGS.
265 REM
27  OPEN "TOOL:BASICARGS" FOR INPUT AS #1
28  INPUT #1, NROWS, NCOLS
285 Dim Del(NRows)
288 True = -1 : False =
29  REM
3   PRINT "Parameters NROWS="; NROWS; "NCOLS="; NCOLS
31  PRINT
32  FOR ROW = 1 TO NROWS
33      REM  DO
34          INPUT #1, T
```

```
35            IF (T =  ) THEN INPUT #1, R
36            IF (T = 1) THEN INPUT #1, R
37            IF (T = 2) THEN INPUT #1, S$ :IF S$ = "Yes" \
               THEN Del(Row) = True ELSE Del(Row) = False
38            IF (T = 3) THEN INPUT #1, R
39            IF (T = 4) THEN INPUT #1, R
4             IF (T = 9) THEN INPUT #1, R, Z
41            REM
42            IF (T =  ) THEN PRINT "Integer:      "; R
43            IF (T = 1) THEN PRINT "Real:         "; R
44            IF (T = 2) THEN PRINT "String:       "; "<"; S$; ">"
45            IF (T = 3) THEN PRINT "Date:         "; R
46            IF (T = 4) THEN PRINT "Boolean:      "; R
47            IF (T = 5) THEN PRINT "Unspecified"
48            IF (T = 9) THEN PRINT "Full date: "; R; Z
49            IF (T = -1) THEN PRINT "N/A"
5             IF (T = -2) THEN PRINT "Error"
51            IF (T = -3) THEN PRINT "--------------------------------"
52        REM UNTIL END OF ROW
53        IF (T <> -3) THEN 33
54  NEXT ROW
55  CLOSE #1
56  REM
57  REM This section copies the input region to the output region
58  REM
59  OPEN "TOOL:BASICIN" FOR INPUT AS #5
6   INPUT #5, NROWS, NCOLS
61  PRINT "Data NROWS="; NROWS; "NCOLS="; NCOLS
62  OPEN "TOOL:BASICOUT" FOR OUTPUT AS #5
63  WRITE #5, NROWS, NCOLS
64  REM
65  FOR ROW = 1 TO NROWS
66      REM  DO
67            INPUT #5, T
68            IF (T =  ) THEN INPUT #5, R
```

```
69          IF (T = 1) THEN INPUT #5, R
7           IF (T = 2) THEN INPUT #5, S$
71          IF (T = 3) THEN INPUT #5, R
72          IF (T = 4) THEN INPUT #5, R
73          IF (T = 9) THEN INPUT #5, R, Z
74      REM
75          IF (T =  ) AND (Del(1) = False) THEN WRITE #5, T, R
76          IF (T =  ) THEN PRINT "Integer:     "; R
77          IF (T = 1)   AND (Del(2) = False) THEN WRITE #5, T, R
78          IF (T = 1) THEN PRINT "Real:          "; R
79          IF (T = 2)   AND (Del(3) = False) THEN WRITE #5, T, S$
8           IF (T = 2) THEN PRINT "String:       "; '<'; S$; '>'
81          IF (T = 3)   AND (Del(4) = False) THEN WRITE #5, T, R
82          IF (T = 3) THEN PRINT "Date:          "; R
83          IF (T = 4)   AND (Del(5) = False) THEN WRITE #5, T, R
84          IF (T = 4) THEN PRINT "Boolean:     "; R
85          IF (T = 5) THEN WRITE #5, T
86          IF (T = 5) THEN PRINT "Unspecified"
87          IF (T = 9) AND (Del(6) = False) THEN WRITE #5, T, R, Z
88          IF (T = 9) THEN PRINT "Full date: "; R; Z
89          IF (T <  ) THEN WRITE #5, T
9           IF (T = -1) THEN PRINT "N/A"
91          IF (T = -2) THEN PRINT "Error"
92          IF (T = -3) THEN PRINT "------------------------------"
93      REM UNTIL END OF ROW
94      IF (T <> -3) THEN 66
95  NEXT ROW
96  CLOSE #5
97  CLOSE #5
98  END
```

2. Save the program–in this example, name it Filter. In the program area of the BASIC tool window, type:

   SAVE "Filter"

   Be sure to include quotes when using the SAVE command.

3. Close the BASIC icon and name it Input.

4. Copy a Transformer icon to your desktop and do the following:
   a. Open the Transformer icon, and select the **Show Controls** button in the window header.
   b. Enter the name of the BASIC program–Filter in this example–in the **Program Name** field.

      The BASIC tool icon and the program name specified in the transformer must be identical.
   c. Close the Transformer icon.
5. Copy two Spreadsheet icons to your desktop, and name them. The Spreadsheet icons should have different names. In Figure 9 on page 28, the input spreadsheet is named Input, and the output spreadsheet is named Results.
6. Open the Spreadsheet icon named Input and, using the spreadsheet setup shown in the following example, create a region and enter data into it.

	A	B	C	D	E	
1	1.00	12.34				
2	2.00	IBM				
3	3.00	Jan 1, 88				
4	4.00	False				
5	5.00					
6	6.00	N/A				
7	7.00	Error				
8						
9						
10						

   Remember that a spreadsheet cell containing a date, as in the previous example, must be defined in the Spreadsheet Options window for the cell as Date.
7. Copy a capsule icon to your desktop, open it, and move the BASIC tool icon, the transformer icon, and the two spreadsheet icons into the capsule window, positioning them as shown in Figure 9 on page 28. For more information about using the capsule tool, refer to the IDS *Capsule User's Guide*.
8. Join the Spreadsheet and Transformer icons using the **Connect Icons** button in the capsule window header.
9. To run the program, click on the **Run** button in the capsule window header.

The BASIC program Filter copies input from the spreadsheet Input to the spreadsheet Results, as shown in the following example.

	A	B	C	D	E	
1	1.00	12.34				
2	2.00	IBM				
3	3.00	Jan 1, 88				
4	4.00	False				
5	5.00					
6	6.00	N/A				
7	7.00	Error				
8						
9						
10						

Spreadsheet results are also written to a BASICLog and are formatted by PRINT statements included in the program. The BASICLog has the same name as the program and appears on the desktop as an icon when the capsule application is finished running.

Chapter 6. Troubleshooting BASIC Programs

If you encounter a problem running your program, use the following troubleshooting guidelines to help resolve it.

Programs for Spreadsheets

Follow these troubleshooting guidelines for BASIC programs written for execution in spreadsheets.

Did you remember to enclose region names in brackets when invoking your program from within a spreadsheet? Region names must be inside square brackets.

Incorrect:	BASIC MYPROG (REG1;REG2)
Correct:	BASIC MYPROG ([REG1];[REG2])

Did you omit a semicolon when invoking your program from within a spreadsheet? A semicolon is required whether or not you specify a region name.

Incorrect:	BASIC MYPROG ([OUTPUT])
Correct:	BASIC MYPROG (;[OUTPUT])

Programs for Transformers

Follow these troubleshooting guidelines for BASIC programs written for execution in transformers.

Did you include REM $$ in front of the following transformer keywords?

INLABELS	ARGLABELS
OUTLABELS	ARGVALUES

Did you include a greater than sign (>) in front of argument labels and argument values?

Programs for Spreadsheets or Transformers

BASIC programs that are written for execution in spreadsheets share some structural similarities with those written for execution with transformers. Follow these trouble-shooting guidelines for both types of BASIC programs.

Did you remember to close one spreadsheet region before opening another? Only one spreadsheet region can be open at a time.

Did you remember to include an end-of-row tag (–3) at the end of each spreadsheet row specified in your program?

© Copyright IBM Corp. 1996

Did you use a PRINT statement instead of a WRITE statement to send output to the spreadsheet? Only a WRITE statement can be used to send output to the spreadsheet.

Did the program attempt to read from more cells than are present in an input region? You cannot specify more input regions than actually exist in a spreadsheet; if you do, the error, Input past end in line_number is sent to the BASICLog icon, where line_number identifies the line in the program where the error occurred.

Did the program attempt to read a cell value that is preceded by one of the following spreadsheet cell tags? These tags (also listed in Table 4 on page 10) do not accept cell values.

–1 (N/A)	–3 (end-of-row)
–2 (error)	+5 (empty cell)

Did the program attempt to read the spreadsheet cell tag (which identifies the data type), cell value, and end-of-row (–3) tag with the same INPUT statement? If the program is reading a value from a spreadsheet region, only the spreadsheet cell tag and cell value can be read with the same INPUT statement. For example:

```
INPUT #5, TYPE, INT2
```

where:

#5	is a variable identifier that you assigned in the TOOL statement to open the input region.
TYPE	is the cell tag that identifies the data type of a spreadsheet column.
INT2	is the cell value.

For more details, see “Reading and Writing Spreadsheet Cell Types and Values” on page 13.

If the program is reading an end-of-row tag from a spreadsheet region, a separate INPUT statement is required. For example:

```
INPUT #5, TYPE          ' Read the end-of-row tag '
```

An example of how the end-of-row tag is read is shown in Figure 2 on page 15.

Does the WRITE statement include both the cell type and cell value? Cell tags and cell values must be written with the same WRITE statement:

```
IF TYPE =
2 THEN WRITE #5,TYPE, VALUE
```

where:

TYPE	is a value whose data type the IBM BASIC interpreter checks; if the condition tests true, the value is written to a spreadsheet output region. In this case, if TYPE is a string, the string is written to output region #5.
2	defines a spreadsheet string type.

#5	is the variable identifier that you assigned to the current output region and that also corresponds to the identifier for the current input region.
VALUE	is a string that you provide.

This is different than with INPUT statements, where cell tags and cell values can be read with separate statements:

```
INPUT #5, TYPE
INPUT #5, VALUE
```

This is described in the section titled, “Reading and Writing Spreadsheet Cell Types and Values” on page 13.

Did the program attempt to use several icons with the same name? If several icons have the same name, only one is checked for the program that you specify in the program run statement. The BASIC interpreter might have run the wrong program.

Appendix A. IBM BASIC Operators and Keyword Syntax

Table 6 lists the BASIC logical, operational, and mathematical operators. Table 7 on page 38 lists IDS BASIC keywords and their syntax. The following syntax conventions apply:

Parameters inside brackets [[ ]] indicate they are optional.

x used as a parameter stands for any numeric expression.

x$ used as a parameter stands for any string expression.

Table 6 (Page 1 of 2). BASIC operators

Syntax	Description
and	logical and
eqv	logical equivalent
imp	logical imply
not	logical not
or	logical or
xor	logical exclusive or
<	relational operator; less than
<=, =<	relational operator; less than or equal to
<>, ><	relational operator; not equal to
=	relational operator; equal to
<=,=>	relational operator; greater than or equal to
>	relational operator; greater than
	math operator; exponentiation
+	math operator; addition
–	math operator; subtraction
*	math operator; multiplication
/	math operator; division
ABS(x)	returns the absolute value of the numeric expression **x**
ASC(x$)	returns a numeric value that is the ASCII code for the first character of the string **x$**
ATN(x)	returns the arctangent of the numeric expression **x**
COS(x)	returns the cosine of numeric expression **x**
EXP(x)	returns an exponent to the power **x**
INT(x)	returns the largest integer less than or equal to **x**
LOG(x)	returns the natural logarithm of **x**
MOD	math operator; returns an integer value that is the remainder of an integer division

© Copyright IBM Corp. 1996

Table 6 (Page 2 of 2). BASIC operators

Syntax	Description
RND[[x]]	returns a random number between 0 and 1
SGN(x)	returns –1 if **x** is negative, 1 if it is positive, or 0 if it is zero
SIN(x)	returns the sine of **x**
SQR(x)	returns the square root of **x**
TAN(x)	returns the tangent of **x**

Table 7 (Page 1 of 6). BASIC keyword syntax

Keyword	Syntax	Description
all	—	see chain
as	—	see open and name
beep	BEEP	beeps
cdbl	CDBL(x)	converts **x** to a double precision number
chain	CHAIN [[MERGER]]filespec [[,[[line_number]] [[,ALL]] [[,DELETErange]]]]	invokes a program with parameters
chr$	CHR$(x)	returns a single-character string whose ASCII code is **x**
cint	CINT(x)	converts **x** to an integer by rounding
clear	CLEAR [[,[[location]] [[,stack]]]]	specifies the highest memory location to be used and sets the stack size
close	CLOSE [[[[#]] file_number [[,[[#]]file_number]]]]	closes specified files
common	COMMON list_of_variables	passes variables to a chain program
cont	CONT	continues program execution after a break
csng	CSNG(x)	converts **x** to a single precision number
cvd	CVD(8-byte-string)	casts string value to double precision float
cvi	CVI(2-byte-string)	casts string value to short integer
cvl	CVL(4-byte-string)	casts string value to long integer
cvs	CVS(4-byte-string)	casts string value to single precision float

Table 7 (Page 2 of 6). BASIC keyword syntax

Keyword	Syntax	Description
data	DATA constant1 [[,constant2]]	stores constants to be used by READ statements
date$	DATE$	retrieves the current date
deftype	DEFtype letter_range	declares all variables within letter_range to be of the specified type
delete	DELETE [[line_number1]] [[-]] [[line_number2]]	deletes program lines
dim	DIM variable(subscripts) [[, variable(subscripts)]]	specifies the upper limit for an array subscript
else	—	see if
end	END	terminates program
eof	EOF(file_number)	tests for end-of-file condition
erase	ERASE array_name[[, array_name]]	deletes array from memory
erl	ERL	returns line number for latest error
err	ERR	returns error code for latest error
error	ERROR code	simulates the occurrence of BASIC error code
field	FIELD [[#]]file_number,field_width AS string	allocates space for variable in file buffer
files	FILES	prints names of files on disk
for	FOR counter =start TO end [[STEPincrement]] . . NEXT [[COUNTER]] [[,counter]]	loops until counter equals end condition
fre	FRE(number) | FRE(string)	returns size of free string space
get	GET [[#]]file_number[[, record]]	reads a record from file to buffer
gosub	GOSUB line_number1 . . RETURN [[line_number2]]	branches to a specified line number and returns when done
goto	GOTO line_number	branches to a specified line number
hex$	HEX$(expression)	returns a string that contains the hexadecimal value of **expression**

Table 7 (Page 3 of 6). BASIC keyword syntax

Keyword	Syntax	Description
if	IF expression [[THEN | GOTO]] target [[:]] ELSE alternative	if **expression**, then specified action; else alternative action
inkey$	INKEY$	returns a keyboard character or null string if no key input
input	INPUT [[;]] [[prompt;]] variable1 [[, variable2]]	reads variables from the keyboard
input#	INPUT[[#]]file_number,variable1 [[,variable2]]	reads variables from file
input$	INPUT$ (x[[, #file_number]])	returns a string of **x** characters from the keyboard or from a file
instr	INSTR([[start_position,]]x$,pattern$)	returns the position of **pattern$** in **x$**
kill	KILL filespec	deletes file from disk
left$	LEFT$(x$, n)	returns a string consisting of the **n** leftmost characters of **x$**
len	LEN(x$)	returns the number of characters in **x$**
let	[[LET]]variable =expression	lets a variable get its value from an expression
line input	LINE INPUT [[;]] [[prompt_string;]] string	inputs an entire line into **string**
line input#	LINE INPUT#file_number,string	inputs an entire line into **string**
list	LIST [[line_number]] [[-[[line_number]]]] [[, device]]	lists a program from memory
load	LOAD filespec [[, R]]	loads a file into memory
loc	LOC(file_number)	returns the current position within the file
lof	LOF(file_number)	returns the length of file in bytes
lpos	LPOS(x)	**x** is always zero, since no printer is currently supported
lset	LSET string =x$	moves data from memory to file buffer
merge	MERGE file_spec	merges a program, or specified file, into the current loaded program

Table 7 (Page 4 of 6). BASIC keyword syntax

Keyword	Syntax	Description
mid$	MID$(x$, n [[,m]])	returns a string of length **m** from string **x$**, beginning with the **n** th character
mkd$	MKD$(double_precision_expression)	casts a numeric value to a string value
mki$	MKI$(integer_expression)	casts a numeric value to a string value
mkl$	MKL$ (long_integer_expression)	casts a numeric value to a string value
mks$	MKS$(single_precision_expression)	casts a numeric value to a string value
name	NAME old_file_name AS new_file_name	changes name of a disk file
new	NEW	clears memory of program and variables
next	—	see **for**
oct$	OCT$(expression)	returns string that represents the octal value of **expression**
on error goto	ON ERROR GOTO line_number	enables error handling
on gosub	ON expression GOSUBline_number_list	branches to a specified subroutine
open	OPEN file_spec[[FOR model]] AS [[#]] file_number[[LEN = reclen]]	open device for input or output
option base	OPTION BASE n	declares the minimum value for array subscripts
pos	POS(x)	returns the current horizontal position of the cursor
print	PRINT #file_number, [[expression_list]]	prints data to a file
print #	PRINT [[USING string_expression;]] [[expression_list]]	prints formatted data to the screen
print using#	PRINT #file_number, [[USING string_expression;]] [[expression_list]]	prints formatted data to the specified file
put	PUT [[#]]file_number[[,record]]	writes a record from buffer to file
randomize	RANDOMIZE [[expression]]	reseeds the random number generator

Table 7 (Page 5 of 6). BASIC keyword syntax

Keyword	Syntax	Description
read	READ variable_list	reads values from DATA statements and assigns them to variables
rem	REM remark	allows remarks in programs
renum	RENUM [[[[new_number]] [[, [[old_number]] [[,increment]]]] [[, [[stop_number]]]]]]	renumbers programs lines
reset	RESET	closes all programs
restore	RESTORE [[line_number]]	allows a DATA statement to be reread
resume	RESUME line_number| RESUME [[0]] | RESUME NEXT	continues program execution
return	—	see **gosub**
right$	RIGHT$(x$, n)	returns a string consisting of the **n** rightmost characters of **x$**
rset	RSET string =x$	moves data from memory to file buffer
run	RUN [[file_number]] | RUN file_spec [[, R]]	executes the program
save	SAVE file_spec[[, A | P]]	saves a program file
spaces$	SPACES$(n)	returns a string of spaces of length **n**
spc	SPC(n)	skips **n** spaces in a PRINT statement
step	—	see **for**
stop	STOP	terminates program execution
str$	STR$(x)	returns a string representation of **x**
string$	STRING$(m, x \| x$)	returns a string of length **m** whose characters all have ASCII code **x** or an initial character of **x$**
swap	SWAP variable, variable	exchanges the value of two variables
tab	TAB(col)	moves the cursor to the specified column
time$	TIME$	retrieves the current time
troff	TROFF	turns trace off
tron	TRON	traces the execution of program statements

Table 7 (Page 6 of 6). BASIC keyword syntax

Keyword	Syntax	Description
val	VAL (x$)	returns numeric value of **x$**
wend	—	see **while**
while	WHILE condition WEND	loops while condition is true
write	WRITE[[expression]	prints data to the screen
write#	WRITE #file_number, [[expression_list]]	prints data to a file

Glossary

BASICIN. A standard name by which a BASIC program identifies a spreadsheet input region for reading input.

BASICOUT. A standard name by which a BASIC program identifies a spreadsheet output region for writing output.

bimonth. A two-month period. An odd bimonth is a two-month period ending in an odd-numbered month–for example, FM. An even bimonth is a two-month period ending in an even-numbered month–for example, JF.

Capsule. A tool for building an automated processing sequence using other icons. Also called a Capsule application.

cell. The smallest data unit in a spreadsheet.

cell tag. A value that identifies a cell type–such as integer, real, or string–in a spreadsheet input or output region.

input region. A range of cells in a spreadsheet that contains input data from another source, such as a BASIC program or from another region in the spreadsheet.

New Icons file drawer. A file drawer that contains the *Public File Drawers and the * Blank Icons file drawer.

options window. Any window that the user can open by using the [Options]key or the options function on the pop-up function pad. Options windows enable the user to view and modify the properties of the selected object.

output region. A range of cells in a spreadsheet that accepts data from another source, such as from a BASIC program or from another region in the spreadsheet.

Special button. A button in some window headers and on the far-right side of the message area. The Special button, which opens a menu, looks like this:

The BASIC tool has no menu choices listed in the Special menu.

Spreadsheet. A tool used to perform calculations and analyses.

Transformer. A tool that can be used to perform various types of data manipulation in a Capsule application together with a Spreadsheet tool and a BASIC program.

© Copyright IBM Corp. 1996

IDS Publications

This section lists IDS publications, including order numbers. To order copies of the books listed here, or to get more information about a book, see your IBM representative.

General Publications

Getting Started with the Intelligent Decision Server (SC26-8892)

Working with the Intelligent Decision Server (SC26-8893)

Installation, Configuration, and Administration Publications

Installing IDS LAN Components (GC26-8877)

System Administration Guide and Reference (SC26-8890)

Messages and Codes (GC26-8878)

Database Gateway Services Publications

Installing Database Gateway Services for UNIX Environments (GC26-8884)

Configuring and Administering Database Gateway Services (GC26-8885)

Administering Databases with IDS (SC26-8891)

Host Services Publications

Planning MVS Host-to-LAN Connectivity (GC26-8880)

Installing Host Services and Configuring Host-to-LAN Communications for MVS (GC26-8881)

Administering Host Services for MVS: DB2 and Cooperative Application Services (GC26-8882)

Developing Applications with Cooperative Application Services (GC26-8883)

Configuring and Administering Mail Gateway Services (GC26-8923)

Open Data Access Service Publications

Developing Applications with Open Data Access Service (SC26-8888)

Data Access Tools Publications

Browser User's Guide (SC26-8900)

Data Entry User's Guide (SC26-8897)

DBA Tool User's Guide (SC26-8894)

SQL Entry User's Guide (SC26-8898)

Data Analysis Tools Publications

BASIC Tool User's Guide (SC26-8908)

Query User's Guide (SC26-8901)

Spreadsheet User's Guide (SC26-8904)

Presentation Tools Publications

Layout User's Guide (SC26-8902)

Plot User's Guide (SC26-8905)

Reporter User's Guide (SC26-8903)

Text User's Guide (SC26-8896)

Application Development Tools Publications

Transformers Guide (SC26-8907)

Capsule User's Guide (SC26-8906)

Forms Design User's Guide (SC26-8899)

Interacting with PC Applications (SC26-8895)

Installing and Configuring IDS Open Clients (GC26-8889)

© Copyright IBM Corp. 1996

Index

Special Characters

\ + \ 6
\Set Block Selection button 2
\Set Line Selection button 2
\Start Capture button 2
> special character 25

A

ARGLABELS keyword for transformers 25, 26
ARGVALUES keyword for transformers 25, 27

B

BASIC environment in IDS 6
BASIC file-control commands 6
  see also commands, unsupported 1
BASIC keywords
  deviations from standard BASIC 5
  for programs written for transformers 25
  not supported in IDS 5
  syntax 38
BASIC operators 37
BASIC tool
  description 1
  environment 6
  general use 3
  using in stand-alone mode 6
  using with spreadsheets 10
  using with transformers 25
  variations between IDS and standard BASIC 5
  window 2, 3
BASIC tool icon, locating 2
BASICIN statement 12
BASICLog window 4
BASICOUT statement 12
bimonth, see even bimonth and odd bimonth 1
Boolean spreadsheet cell tag 10

C

capsule, running a BASIC program in 25
capturing program runtime results 4
cell data types and values for spreadsheets 10
cell tags, see spreadsheet cell tags 1
clearing the working buffer in stand-alone mode 6
CLOSE statement 12
closing a spreadsheet region 12
commands, BASIC file control 6
  see also commands, unsupported 1
commands, unsupported 5
CONTINUE command 6
conventions, see BASIC program conventions and syntax 1
copying a BASIC program to a BASIC tool window 7
copying a BASIC program to a BASIC tool window 3

D

data transmission, tracing 24
data types, for spreadsheets 10
date resolution codes for spreadsheets 11
date spreadsheet cell tag 10
deleting a program 6
differences between IDS BASIC and standard BASIC 5
dimensions of a spreadsheet region 12, 13, 14, 15
displaying program contents 6

E

end-of-row spreadsheet cell tag 14
end-of-row spreadsheet cell tag 10, 14
entering a program in the BASIC tool window 7
entering values to a BASIC program 7
even bimonth 11
executing a program, see running a program 1

F

file-control commands 6
FILES command 3, 6
functions, unsupported 5

I

INLABELS keyword for transformers 25, 26
input region
  defined 10
  finding dimensions of 14
  in a RUN statement 22
  opening and closing 12

© Copyright IBM Corp. 1996

integer spreadsheet cell tag 10
invoking a program, see running a program 1

K

keywords
see BASIC keywords 1
KILL command 6

L

line continuation 6
LIST command 3, 6, 8
LOAD command 6, 7
loading a program in the BASIC tool 7
locating a BASIC icon 2
logging a session 4

M

modifying a BASIC program 8
multiple input and output regions, specifying in RUN statement 22

N

NAME command 6
naming conventions for BASIC programs 6
NEW command 6
new date spreadsheet cell tag 10, 11
null spreadsheet cell tag 10

O

odd bimonth 11
ON ERROR GOTO statement 24
OPEN statement 12, 13, 24
opening a spreadsheet region 12
operators 37
OUTLABELS keyword for transformers 25, 26
output region
defined 10
finding dimensions of 13, 15
in a RUN statement 22
opening and closing 12

P

PRINT statement 33
printing program results 9
program area 3
program area in BASIC tool window 2
program conventions and syntax
spreadsheets 10, 15
stand-alone mode 6
transformers 25, 27
program naming conventions 6
program sample
for a spreadsheet 19
for a transformer 28
prompt labels in programs for transformers 25

R

READ statement 10, 12
real spreadsheet cell tag 10
recalculating spreadsheet values 23
recording a session 4
REM $$ special character 25
renaming a program 6
restarting a program 6
RUN command 3, 6
RUN statement, in a spreadsheet 16, 17
run-time errors, trapping 24
running a program
from a spreadsheet in a capsule 10, 17, 23
in stand-alone mode 6
scenarios for 16
with a transformer in a capsule 25, 27, 33

S

sample program
for a spreadsheet 19
for a transformer 28
SAVE command 6
saving a program 6, 32
Show Document Controls button 4
single input and output regions, specifying in RUN statement 22
special characters for transformers
> 25
REM $$ 25
spreadsheet
BASIC program conventions and syntax for 10, 15
cell tags 10

spreadsheet *(continued)*
date resolution codes 11
READ statement in BASIC programs 10
recalculating values 23
running a program from 17, 23
sample program for 19
WRITE statement in BASIC programs 10
spreadsheet cell tags 10
spreadsheet region, see input region and output region 1
spreadsheet values, recalculating 23
stand-alone mode
BASIC environment 6
conventions 6
stopping a program 6, 23
string spreadsheet cell tag 10
syntax conventions, see BASIC program conventions and syntax 1

T

TOOL statement 12, 15
tracing data transmission between spreadsheet regions 24
transformer
running a program with 27, 33
sample program for 28
writing a BASIC program for 25
trapping run-time errors 24
troubleshooting tips 34, 36

U

unsupported commands and functions 5

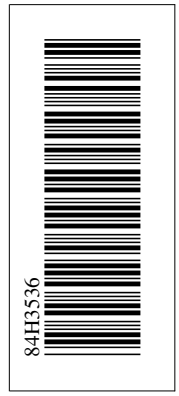
V

variations between IDS BASIC and standard BASIC 5

W

window buttons 2
WRITE statement 10, 13
writing a BASIC program
for a spreadsheet 10, 15
for a transformer 25, 27
writing program results 8

IBM®

Part Number: 84H3536

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.