Reality v10.0

DataBasic Quick Reference Guide

NECSWS.COM

\Orchestrating a brighter world

Document control

Software Version	Document Status	Document Revision	Issue Date	Reason for Change
v10.0	Published	v0.1	June 2003	Final draft

Table of Contents

Section 1: About this guide

1.1 Purpose

This guide summarises all DataBasic elements, including statements, intrinsic functions and debugger commands. It summarises the syntax, function, restrictions, parameters and options of each. Associated TCL commands are listed.

It will be useful to experienced programmers and to those learning to program in DataBasic. Knowledge of Reality or Pick fundamentals is essential to write effective DataBasic programs.

Note
It is outside the scope of this guide to fully define each element in context. If in doubt, check the relevant detailed description in the DataBasic Reference Manual.

1.2 References

DataBasic Reference Manual

This gives detailed descriptions of all DataBasic elements, with examples of their use.

EDITOR Reference Manual

SCREEN EDITOR Reference Manual

These describe utilities used to create and modify DataBasic source code items.

English Reference Manual

This includes detailed descriptions of conversion codes that can be used in DataBasic (Appendix C of this guide summarises them).

1.3 Element Descriptions

For each element in the following chapters, details are given (where applicable) about:

- General Syntax

  Conventions are defined at the end of this chapter.
- Function and Restrictions

  Briefly describes the effect of using the element and any restrictions in its use.
- Special Parameters

  Defines special parameters shown in general syntax. (Frequently-used parameters are defined below.)

Chapter 3 describes all valid statements and functions. Each element in that chapter that is not marked FUNCTION is a statement.

1.4 Frequently-Used Parameters

The following definitions apply except where otherwise indicated.

account

Name of account if different from current account.

array

Name of dimensioned array whose elements are assigned to a file item; must be a vector or a two-dimensional array.

attr#

Specifies location of attribute within array. Command is performed on whole attribute if you specify a nonzero value for *attr#* and zero for both *value#* and *subvalue#*.

data-sect
Name of data section if different from file name. It must follow the file name after a comma.
DICT
Specifies dictionary section of file.
dyn-array
Dynamic array for command to act on.

ELSE

Precedes statement(s) to execute if condition is false or statement execution fails.

expr

Any valid DataBasic expression or any string, substring, or value; expressed as a variable name, a value, or a string enclosed in quotes.

file-var

Variable to which name of file was assigned via an OPEN statement. If not specified, internal default filevariable is used, that is, file most recently opened without a file-variable.file-modifier.

filespec

Defines the file that the command or statement will operate on. It has the syntax:

{**DICT**} {/*account*/}*file*{,*data-sect* }

index-name
Name of an index.

label

Numeric or alphanumeric label of statement.

List name

name of a list, expressed as a variable name or a literal enclosed in quotes.

LOCKED

Precedes statement(s) to execute if item is locked.

mins

Specifies minutes before TIMEOUT. ELSE clause is executed if the connection is not made within this time.

ON ERROR

Equivalent to **ELSE** clause (precedes statement(s) to execute if condition is false or statement execution fails).

sess-var

Variable defined by **CONNECT** and **ACCEPT** statements that identifies session. If omitted, uses/establishes default session.

setting-var

Name of variable that will be set to value corresponding to:

- A code by the **SETTING** clause if attribute cannot be written and **ON ERROR** clause is taken. If **ON ERROR** clause is not taken, *setting-var* is set to 0.
- An error code by **SETTING** clause if **ELSE** clause is taken. If **ELSE** clause is not taken, *setting-var* is set to 0, except for **MATREAD** statement which sets it to the number of attributes read, **SELECT** statement which sets it to the number of items in the list created and GET-LIST and READ-LIST which both set it to the number of items in the list.

stmnt(s)

Any number of valid DataBasic statements (eg in **ELSE** or **ON ERROR** clauses), either separated by semicolons or contained on separate lines and followed by **END** statement.

string

Strings and substrings etc can be variable name, dynamic array reference, or literal enclosed in quotes.

subval#

Location of subvalue within value. Command is performed on specified subvalue if you specify nonzero values for *attr#*, *val#* and *subval#*.

THEN

Precedes statement(s) to execute if condition is true or statement execution succeeds.

tran-info

Optional text to be saved in transaction-start record. Can be used to identify a particular transaction or iteration of a repetitive transaction.

value#

Specifies location of value within attribute. Command is performed on value if you specify a nonzero value for both ***attr#*** and ***value#*** and a zero for ***subvalue#***.

1.5 Conventions

Example	Meaning
TEXT	Bold text represents characters typed exactly as shown.
{Braces}	Braces enclose options and optional parameters.
Document Title	Italic text also indicates titles of documents referenced.
Text	Characters or words in italics indicate parameters which must be supplied by the user.
[param | param]	Parameters shown separated by vertical lines within square brackets in syntax descriptions indicate that at least one of these parameters must be selected.
CTRL+X	Two (or more) key names joined by a plus sign (+) indicate a combination of keys, where the first key(s) must be held down while the second (or last) is pressed. For example, CTRL+X indicates that the CTRL key must be held down while the X key is pressed.
RETURN	Small capitals show key names.
X'nn'	This denotes a hexadecimal value.

Section 2: Elements

2.1 Program Format

Programs consist of valid statements, comment lines and blank lines. Multiple statements can be included in a line separated by semicolons. Programs are stored as file items and referred to by item-id. Compiled versions of programs are created by **BASIC** command and are stored as items in the same file as source item, but with item-id preceded by £ (or $). Each line within the program is an attribute of that item.

2.2 Labels

Any statement can be preceded by a label consisting of either a number, or of a letter followed by other characters; alphanumeric labels preceding statements are terminated by a colon. One or more spaces separate the label from the rest of the line. References within other statements to alphanumeric labels do not include the colon.

2.3 Comments

Comments can be included on separate lines or as a separate statement following a semicolon. Every comment must be preceded by **REM**, ***** or **!**. These have different effects when the program is listed via **BLIST**.

2.4 Line Continuation

Lines can be continued with an ellipsis (...) at end of a line.

2.5 Variables

Variables must have names consisting of a letter followed by any printable characters except commas or hyphens. Reserved words cannot be used (see Appendix B).

2.6 Operator Precedence

When more than one operator appears in an expression, operators are processed in the following order: arithmetic; format string; concatenation; relational; logical. Operators of equal precedence are processed from left to right.

2.7 Arithmetic operators

Arithmetic operators are used with numeric values (constants, variables or intrinsic functions with numeric results) to create numeric expressions. They are (in order of evaluation)

(expression)	an expression within parentheses
^	an exponent
+ -	sign (positive or negative)
* /	multiplication and division

+ -	addition and subtraction

2.8 Format Strings

Values can be formatted using format strings. Multiple format strings can be applied to a value and are processed from left to right. The general form is:

"{*j* } { **£**} {,} {*n*} {*field* }"

or

"*output conversion*"

j

L for left or R right justification (default is **L**).

£

Character is put in front of value (or **$** instead).

,

Puts comma between each three digits before decimal point.

n

Specifies number of decimal places to include (**1** to **9**).

field

to 'pad' field:

#{#}... or #r with spaces

%{%}... or %r with zeros

{}... or *r with asterisks

where the number of pad characters is typed (## for two spaces) or where r is a number specifying total field width (excesss characters are truncated).

output conversion

any conversion valid in an **OCONV** function.

2.9 Concatenation

Strings can be concatenated using colon (:) or **CAT** operator placed between them.

2.10 Relational operators

Relational Operators compare two expressions to give the result 1 if relation true, 0 if relation false. They are (in order of evaluation):

= or EQ	Equal to
< or LT	Less than

> or GT	Greater than
<= or LE	Less than or equal to
>= or GE	Greater than or equal to
# or NE	Not equal to
MATCH{ES}	pattern matching

Pattern Matching

One of the operands in a **MATCH** expression defines a pattern. Its general form is:

" {*n***N**} {*n***A**} {*n***X**} {'*string*'} {**]**}..."

where **]** is a value mark (X'FD' or CTRL+]).

Each pattern consists of one or more of the elements shown, in any order: *n***N** (an integer followed by **N**) matches n numeric characters; *n***A** matches in alphabetic characters; *n***X** matches n alphanumeric characters; 'string' matches that literal string. A number of patterns can be specified, separated by value marks: if the first operand matches any pattern the result is true. The whole of the pattern operand is enclosed in double quotes. 0**N** matches any number of numerics, including none (a null string). Similarly, 0**A** and 0**X** match any number of alphabetic or alphanumeric characters respectively.

2.11 Logical operators

AND or &	logical AND
OR or !	logical OR

Logical operators compare relational and arithmetic expressions to give the result 1 (if operator is **OR** and either expression is non-zero, or if operator is **AND** and both expressions are non-zero) or 0.

2.12 Substring Extraction

Substring extraction statements assign elements extracted from a string to a variable.

X = *string*[*start#*,*length*]

X = *dyn-array* {<*attr#*{,*val#*{,*subval#* }}>}

{[*start#*,*length*]}

X = *array* (*row* {*col* }) {<*attr#* {,*val#*{,*subval#* }}>} {[*start#*,*length*]}

start#

starting character position (if <0, count from right).

length

length of substring (if <0, end of substring is *length* from first character to right of string, eg, -1 extracts from *start#* to end of string).

2.13 Substring Assignment

Assignment statements assign a value to a substring.

X[*start#*,*replace#*]=*expr*

X{<*attr#*{,*val#*{,*subval#* }}>}{[*start#*,*replace#*]}=*expr*

X(*row* {,*col* }){<*attr#*{,*val#*{,*subval#* }}>}{[*start#*,*replace#*]}=*expr*

2.14 Dimensioned Arrays

These must be declared via a **DIM**{**ENSION**} or **COMMON** statement before use.

- A one-dimensional array (vector) has one column of elements.
- A two-dimensional array (matrix) has rows and columns of elements.

Elements are accessed by specifying their position: for instance, **A**(**2**) is the second element in vector **A**, and **B**(**2**,**4**) is the element in row 2, column 3 of matrix **B**.

Each element can contain a numeric or a string value.

2.15 Dynamic Arrays

A dynamic array consists of attributes separated by attribute marks (X'FE', often represented by ^).

Attributes can contain values separated by value marks (X'FD', represented by]).

Values can contain subvalues separated by subvalue marks (X'FC', represented by \).

Elements of dynamic arrays are referenced via the attribute, value and subvalue numbers as follows:

dyn-array {<*att#*{,*val#*{,*subval#* }}>}

If the position specification is omitted, the whole array is referenced. If just *att#* is specified, the whole of that attribute is referenced (and so on).

Segment marks (X'FF') are used by system as terminators. Strings containing segment marks should not be used in dynamic array operations.

2.16 Locks

Locks can be set dynamically at various levels to restrict access to resources. Note these locks are entirely distinct from file access and retrieval locks, which are set to protect files against unauthorised access and remain in effect until changed via an editor.

Relevant TCL commands are listed in the TCL Quick Reference Guide, Appendix A.

Read, Update and System Locks

These are used internally by the system.

Execution Locks

These prevent DataBasic or Proc programs executing simultaneously if they set the same lock.

Locks are set by **LOCK** statement in DataBasic or **PL***n* in Proc; another program cannot set the same lock until the first program unlocks it (by **UNLOCK**, terminating program or logging off).

Item Locks

These ensure only one process at a time can access a locked item. Note that an item lock that is set more than once must be released an equal number of times to free the lock.

In DataBasic, locks are set by statements **READU**, **READVU**, **MATREADU** and **READ** with a **LOCKED** clause, and cleared by **RELEASE**, **WRITE**, **WRITEV** or **MATWRITE** or by terminating program, logging off or using CTRL+BREAK followed by END. **TRANSEND** and **TRANSABORT** also clear locks set within the preceding transaction.

WRITEU, **WRITEVU** and **MATWRITEU** allow user to write to an item without unlocking it.

In Proc, locks are set by **F-UREAD** and cleared by **F-F**{**REE**} or **F-WRITE**, or by execution of TCL command **TRANSABORT** or **TRANSEND** (to clear locks set within the preceding transaction). An item is also locked when being updated by the line or screen editor.

2.17 Transactions

Transaction handling is a standard feature which groups updates together as a single transaction. If the entire transaction cannot be completed, updates to files within it are 'rolled back'. Item lock release is delayed until the end of each transaction. To avoid 'deadly embrace', lock and unlock the same set of items in the same sequence.

A set of updates can be identified as a single transaction using **TRANSTART**, **TRANSEND** and **TRANSABORT** statements; **TRANSQUERY** is a function that returns the transaction status of the current port. These work within DataBasic (or from Proc, ALL or RPL).

Transaction logging is an optional feature which saves all updates (or updates to specified account and/or files) to disk.

Section 3: Statements and Intrinsic Functions

3.1 $CHAIN Statement

$CHAIN *item* {**FROM** *filespec*}

or

$CHAIN *filespec item*

Allows two DataBasic source code modules to be compiled as a single program. Cannot be used with source compacted programs.

item

item-id of source code item to chain.

filespec

file containing item to chain. If omitted, item is retrieved from file containing item being processed.

3.2 $OPTIONS {tag}

Set the compatibility mode for the current code module.

tag

One of the following string values, representing the required MultiValue system:

"REALITY"	Reality
"EXT"	Reality (extended)
"PICK"	Pick/Raining Data R83
"R83"	Pick/Raining Data R83
"AP"	Pick/Raining Data, Advanced Pick
"D3"	Pick/Raining Data D3
"GA"	General Automation, GA Pick
"PWR95"	General Automation, Power 95
"INFORMATION"	Prime Information
"IN2"	Prime Information IN2

The default is "REALITY".

3.3 @ Function

@(*col-val* {,*row-val* })

or

@ (*code*)

Sets cursor to specified position on terminal and generates video effects characters.

Syntax elements	Description
col-val	sets cursor to specified column on current line.
row-val	different line number on which to position cursor.
code	generates extended cursor addressing code or video effects. Cursor addressing codes are:

-1	Clears screen.
-2	Cursor home.
-3	Clears to end of screen.
-4	Clears to end of line.
-9	Cursor back.
-10	Cursor up.
-11	Cursor on.
-12	Cursor off.
-13	Status line on.
-14	Status line off.
-15	Cursor forward.
-16	Cursor down.
-17	Slave port on.
-18	Slave port off.
-19	Screen dump.

3.4 A

ABORT {*msg-id* {,*msg-expr*}...}

Halts program execution, prints optional message, and terminates a driving Proc.

Syntax elements	Description
msg-id	item-id of item in system message file (ERRMSG) containing the message; must be numeric.
msg-expr	variable, function, arithmetic statement or literal string that can be printed as part of message. They are processed on a first-in first-out basis.

ABS(*expr*)

Generates absolute (positive) numeric value of expression.

ACCEPT *accept-str* {**TO** *sess-var*} {**TIMEOUT** *mins*} {**SETTING** *setting-var*} {**RETURNING** *client* } [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Establishes session for data exchange with a communicating program that has executed a **CONNECT**.

accept-str

string in the form:

{***PTP***}*server*

or

TCP*host* ; **port=***port-num* {;*option*}{,*option*}...

where:

PTP

specifies Reality process-to-process connection (default).

server

string expression identifying this program for connection requests.

TCP

specifies raw TCP/IP connection.

host

IP address of local network interface.

port-num

port number on host.

option

name/value pair (separated by equals sign), specifying optional parameter.

client

variable that is assigned dynamic array with two attributes:

*client-PLId^client-system*user-id*

ACCESS(*data-element*)

Returns current state of data elements. Use only in file trigger subroutines.

Syntax elements	Description	
data-element	Number corresponding to data element referenced (those omitted are not currently used):	
	1	Reference to trigger file.
	2	If trigger file is a dictionary, reference to trigger file. If trigger file is data section, reference to dictionary of trigger file.
	3	Item body. Null if delete operation.
	10	Id of item being written or deleted.
	11	File name: {**DICT**} {/*account*/}*filename*{,*data-section-name*}.
	12	True if **PRE-DELETE** or **POST-DELETE** trigger.
	13	Always returns 0.

Syntax elements	Description	
	16	True if new item; false if existing item.
	20	In **POST-WRITE** trigger, if true, indicates that item was modified by PRE-WRITE trigger; if false, item was written without modification. Always false in **PRE-WRITE**, **PRE-DELETE** and **POST-DELETE** triggers.
	23	Calling environment. Currently always 1 (trigger).

ALPHA(*expr*)

Searches string for alphabetic characters.

ASCII(*expr*)

Converts string value from EBCDIC to ASCII.

ASSIGN *set-value* **TO SYSTEM**(*sys-element*)

Assigns value to some system and data elements whose values can be retrieved using **SYSTEM** function.

Syntax elements	Description
set-value	Value to which system element is to be set.
sys-element	Number corresponding to system element; valid values are: **2**, **3**, **5**, **7**, **30**, **35**, **37**, **38** or **39** (see **SYSTEM** function).

3.5 B

BCC(*string*)

Generates Binary Check Character (BCC).

Syntax elements	Description
string	variable to which data string has been assigned or literal string enclosed in quotes.

BITCHANGE(*bit-val*)

Toggles state of the specified bit in bit table and returns the value of the bit before it was changed.

Syntax elements	Description
bit-val	bit to change.

BITCHECK(*bit-val*)

Returns current value of specified bit from bit table.

Syntax elements	Description
bit-val	bit to check.

BITLOAD({*bit-string*})

Assigns values to entire bit table or retrieves current value of entire table.

Syntax elements	Description
bit-string	ASCII string representing hex value. Used as bit pattern to assign values to table from left to right. Assignment stops when string runs out or when non-hex character is encountered. If string defined less than 128 bits, remaining bits in table are reset. If bit-string is omitted or evaluates to null, an ASCII hex character string is returned, which defines value of the table. Any trailing zeros are truncated.

BITRESET(*bit-val*)

Resets value of specified bit in bit table to 0 and returns the value of the bit before it was changed.

Syntax elements	Description
bit-val	bit to reset. If bit-val evaluates to zero, all elements in table are cleared and returned value is zero.

BITSET(*bit-val*)

Sets value of specified bit in bit table to 1 and returns the value of the bit before it was changed.

Syntax elements	Description
bit-val	bit to set. If bit-val evaluates to zero, all elements in table are cleared and returned value is zero.

BREAK {**KEY**} [**ON** | **OFF**] or **BREAK** *expr*

Enables or disables BREAK key on terminal.

Syntax elements	Description
expr	evaluates to numeric value. If zero, disables BREAK key; if nonzero, enables BREAK key.

3.6 C

CALL *ctlg-id* {(*argument-list*)}

or

CALL *@ctlg-var* {(*argument-list*)}

Transfers control to external subroutine.

Syntax elements	Description
ctlg-id	external subroutine which is compiled and cataloged separately from program(s) that calls it.
ctlg-var	variable containing name of cataloged subroutine to call.
argument-lis	one or more expressions, separated by commas, representing actual values passed to subroutine. Called subroutine must have same number of items in its argument-list, listed in same order, (if not, an error message is displayed and program enters debugger)

BEGIN CASE

CASE *expr*

stmnt(s)

CASE *expr*

stmnt(s)

END CASE

Allows conditional selection of sequence of statements. If first expression is true (nonzero), statement(s) that immediately follow are executed and control passes to next sequential statement following **END CASE**. If first expression is false (zero), then control passes to next test expression and so on.

Syntax elements	Description
expr	expression that evaluates to true (1) or false (0).

CHAIN *cmd*

Allows DataBasic program to exit to any TCL command or passes values to separately compiled programs.

Syntax elements	Description
cmd	any valid TCL command, cataloged DataBasic program or Proc in user's MD.

CHANGE(*old-str*, *old-substr*, *new-substr*)

Replaces substring. Each string must be in quotes or given by a variable name.

Syntax elements	Description
old-str	original string.

Syntax elements	Description
old-substr	substring in original string to change.
new-substr	replacement substring.

CHAR(*char-val*{,*size*})

Converts char-val to its corresponding ASCII character string value as one or size charaters (size is 1 to 4 bytes).

Syntax elements	Description
char-val	positive integer in range **0** - **255**, representing decimal value of an ASCII character.

CHECKSUM(*expr*)

Returns number equal to checksum of specified string.

CLEAR

Initialises all program variables to zero.

CLEARFILE {*file-var*} {**SETTING** *setting-var*} {**ON ERROR** *stmnt*(*s*)}

Clears specified file. D-pointers are not deleted. SYS2 privileges required.

CLOSE *filevar* {,*filevar*} ...

Used when you no longer need to access an **OPEN**ed local or remote system file.

COLLECTDATA *var*

Retrieves data passed by **PASSDATA** clause of **PERFORM** statement.

COL1()

COL2()

Returns numeric values of column positions immediately preceding and following substring specified by **FIELD** function.

COMMON {/ *common-name* /} *var* {,*var* }...

Defines variables shared by programs in specified common area. Different variable names can be used in each program, but must be defined in the same order. Arrays are specified by declaring dimensions (in parenthesis) after array name. Arrays in **COMMON** statements should not be declared in a **DIMENSION** statement.

If *common-name* is omitted, local common (shared between current program and its subroutines) is used. Named common sections are shared between all programs run in a logon session.

Syntax elements	Description
common-name	named common section.
var	any simple, dimensioned or file variable.

CONNECT *connect-str* {**TO** *sess-var*} {**TIMEOUT** *mins*} {**SETTING** *setting-var*} [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Establishes connection for data exchange with a communicating program.

connect-str

string in the form:

{***PTP***}{*system*}^{*acct* {,*acct-pswd* }}^*server*{,*user-pswd* }{**^Q**}

or

TCP*host* ; **port=***port_num* {;*option*}{,*option*}...

where:

PTP

specifies Reality process-to-process connection (default).

system

Entry in /etc/ROUTE-FILE or Windows registry for remote system: default is local database.

^

Attribute mark – CHAR(**254**).

acct

Account on which server is to be started (if not already running).

acct-pswd

Password for account.

server

Command in acct's MD which executes server program, or name of server program already running.

user-pswd

Password for user-id server on remote system (if it exists and needs a password).

Q

Queues connect request until already running server issues an ACCEPT.

TCP

specifies raw TCP/IP connection.

host

IP address or DNS domain name of remote system.

port_num

port number on host.

option

name/value pair (separated by equals sign), specifying optional parameter.

CONVERT(*string*, *old-substr*, *new-substr*)
Replaces individual characters within a string.

Syntax elements	Description
string	original string.
old-substr	list of characters to change.
new-substr	list of replacement characters.

COS(*expr*)

Calculates cosine of an angle.

Syntax elements	Description
expr	expression giving angle in degrees: 2 Pi radians = 360 degrees

COUNT(*string*,*substring*)

Counts number of times substring occurs within string (substrings may overlap).

CRC(*string*{,*type*})

Generates Cyclic Redundancy Character (CRC).

Syntax elements	Description
string	variable to which data string has been assigned or literal string in quotes.
type	type of polynomial used to generate CRC: 1 CRC-CCITT (X.25 Standard) 0 CRC16 Default value is 0.

CRT {*print-list* }

Outputs data to terminal. Similar to **PRINT**, but always outputs to CRT.

Syntax elements	Description
print-list	single expression or series of expressions, separated by commas or colons. They may be any text string enclosed in quotes, numerical value, variable that evaluates to a text string, or expressions used to denote output formatting (including format strings). See also the **@** function.

3.7 D

DATA *data-string*{,*data-string*}...

Stores values used by subsequent requests for terminal input due to **INPUT**, **CHAIN** or **PERFORM** statement.

Syntax elements	Description
data-string	any data to be stored to satisfy subsequent requests for input, expressed as variable or literal in quotes. Each string is queued as one line of input. Subsequent requests for input are met on a first-in-first-out basis. Each string is limited to 240 characters.

DATE()

Returns string value containing internal system date.

DCOUNT(*string*,*delim*)

Counts number of elements in string that are separated by specified delimiter.

Syntax elements	Description
delim	delimiter expressed as literal in quotes or variable name.

DEBUG

Passes control to DataBasic Symbolic Debugger.

DECRYPT(*exp1*,*exp2*,*method.idx*)

Decodes string variable that was previously **ENCRYPT**ed.

Syntax elements	Description
exp1	plain text or encyphered text string to decrypt.
exp2	string that is decrypt key (the same as that used for encryption).
method.idx	method of decryption; it can be: 0 General purpose (uses last character of exp2). 1 Rotation algorithm (ROT13) affecting only alphabetic characters; exp2 can be null ("").

Syntax elements	Description
	2 XOR.MOD11 algorithm; exp2 must be single-character. 3 One-for-one exclusive OR between exp1 and infinite garbage string (whole of exp2 used).

DEL *dyn-array<attr#{,val#{,subval# }}>*

Deletes attribute, value or subvalue from dynamic array. (Supersedes the intrinsic function **DELETE**.)

Syntax elements	Description
attr#	attribute within referenced dynamic array.
val#	value within referenced attribute.
subval#	subvalue within referenced value.

DELETE (*dyn-array,attr#* {*,val#* {*,subval#* }})

Deletes attribute, value or subvalue from dynamic array. This function is superseded by the DELstatement but is maintained for compatibility.

DELETE {*file-var,*}*item-id* {**SETTING** *setting-var*} {**ON ERROR** *stmnt*(*s*)}

Deletes file item.

DELETELIST *list-name* {*account* }

Deletes previously saved list from POINTER-FILE.

Syntax elements	Description
account	to delete lists saved from another account.

DIM{**ENSION**} *array*(*dim*) {*,array*(*dim*)}...

Dimensions arrays for use in DataBasic program. See also **COMMON** statement.

Syntax elements	Description
array	name of array.
dim	size of array. Array can be one or two-dimensional (vector or matrix). Dim is specified as (r) or (r, c) respectively, where r is number of rows, c is the number of columns.

DISCONNECT {*sess-var*} {**SETTING** *setting-var*} [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Terminates network session.

DOWNCASE (*expr*)

Returns *expr* with all uppercase letters converted to lower case (like conversion MCL).

DQUOTE(*string-var*)

Returns specified string surrounded by double quotes.

Syntax elements	Description
string-var	field of characters expressed as variable.

DTX(*expr*)

Converts a decimal value to hexadecimal.

3.8 E

EBCDIC(*expr*)

Converts string value from ASCII to EBCDIC.

ECHO ON or **ECHO OFF** or **ECHO** *expr*

Controls echoing of input characters.

Syntax elements	Description
expr	variable or expression that evaluates to zero (echoing is disabled) or non-zero (echoing is enabled).

ENCRYPT(*exp1*,*exp2*,*method.idx*)

Encodes string variable.

Syntax elements	Description
exp1	plain text or encyphered text string to encrypt.
exp2	string that is encrypt key.
method.idx	method of encryption; see DECRYPT for details.

END

Marks end of DataBasic program (optional). Also ends multi-line IF, THEN, ELSE and ON ERROR clauses.

ENTER *item-id* or **ENTER** @*ctlg-var*

Transfers control from one cataloged DataBasic program to another.

Syntax elements	Description
ctlg-var	variable that evaluates from one name of cataloged program to another.

EQU{**ATE**} *symbol* **TO** *relation* {,*symbol* **TO** *relation*}...

Declares symbol to be equivalent to variable or literal.

Syntax elements	Description
symbol	formed like a variable, but no storage is allocated for it. Cannot be a reserved word (see Appendix B).
Relation	number, literal string, character, simple variable, array element or CHAR function to equate to symbol. If *relation* is simple variable, the two variable names are equivalent and can be used interchangeably.

EXP(*expr*)

Raises 'e' to specified value.

EXTRACT(*dyn-array*,*attr#* {,*val#* {,*subval#* }})

Returns attribute, value or subvalue from dynamic array.

3.9 F

FIELD(*string*,*delim*,*delim-occur*)

Returns substring from within string.

Syntax elements	Description
delim	character marking end of substring to be returned.
delim-occur	integer delineating which appearance of delimiter is used to mark substring.

FIND *loc-field* **IN** *dyn-array*{,*occur*} **SETTING** *attr-var*{,*val-var*{,*subval-var*}} [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Locates position of given attribute, value or subvalue in dynamic array. If element not found, ELSE is executed and SETTING variables are not changed.

Syntax elements	Description
loc-field	string or value being searched for, expressed as variable name, numeric constant, or literal in quotes.
occur	occurrence number of element being searched for (default is first occurrence).
attr-var	set to attribute position where element is found.
val-var	set to value position where element is found.

Syntax elements	Description
subval-var	set to subvalue position where element is found.

FINDSTR *string* **IN** *dyn-array*{,*occur*} **SETTING** *attr-var*{,*val-var*,{*subval-var*}} [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Locates substring within dynamic array element. If string not found, ELSE is executed and SETTING variables are not changed.

Syntax elements	Description
string	string to search for within element in dynamic array; expressed as variable name, numeric constant, dynamic array reference, or literal enclosed in quotes.
occur	occurrence number of element being searched for (default is first occurrence).
attr-var	set to attribute position where element is found.
val-var	set to value position where element is found.
subval-var	set to subvalue position where element is found.

FMT (*x*, *y*)

Applies a mask character conversion to a variable.

Syntax elements	Description
x	Variable or input string to be converted.
y	MC (mask character) conversion (see Appendix C).

FOLD (*string*,*fold-width*)

Places attribute marks in a string in place of spaces no more than fold-width apart.

Syntax elements	Description
string	any text containing spaces.
fold-width	maximum number of characters in each 'fold'.

FOOTING *expr*

Pages current output device and prints specified text at bottom of page. See HEADING for special control characters.

FOR *var*=*init* **TO** *test* {**STEP** *inc*} …

{[**WHILE** | **UNTIL**] *limit* }

{ *stmnt*(*s*) }

{

{[**WHILE** | **UNTIL**] *limit* {**DO**}

{ *stmnt*(*s*) }}

}...

NEXT *var*

Used to construct loops. **FOR** begins loop; **NEXT** marks end of loop. Loops can be nested, and **WHILE** or **UNTIL** clauses can be included anywhere within the loop. **DO** has no effect on execution.

Syntax elements	Description
var	variable incremented or decremented by **NEXT** statement. Same var used in **FOR** and **NEXT** statements delimiting loop.
init	initial value of variable.
test	limiting value of variable.
inc	number by which to increment *var*. Default is 1. inc
limit	expression that evaluates to true (1) or false (0).

GETLIST *list* {*account* } {**TO** *list-var*} {**SETTING** *setting-var*} [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Gets item-id list for subsequent **READNEXT** or **PERFORM**. **ELSE** clause executed if list not in POINTER-FILE.

Syntax elements	Description
list	variable or expression giving name under which list was saved.
account	specified if list saved from another account.
list-var	variable to which list is assigned. If **TO** clause omitted, default list variable is used.

GETMSG(*class#*,*msg#*)

Retrieves messages from system denationalization language tables.

Syntax elements	Description
class#	message class number in language table.
msg#	message number within class in language table.

GOSUB *label*

Transfers control to subroutine with specified label.

GO{**TO**} *label*

Unconditionally transfers program control to statement with specified label.

GROUP(*string*,*delim*,*start-grp*,*rtrn-grp*) *FUNCTION*

Returns set of substrings from string. Similar to **OCONV** with code **G**, but *delim* can be system delimiter.

Syntax elements	Description
string	variable or literal in quotes in which substring located.
delim	character that separates each group in string.
start-grp	number of first group to return. Note that this is different from **OCONV** function, which specifies number of leading groups to skip.
rtn-grp	number of groups to return. If more than groups in string, groups are returned until string exhausted.

GROUPSTORE *string1* **IN** *string2* **USING** *start#*, *replace#* {,*delim-char*}

Inserts group (substring delimited by an attribute mark or other specified character) from one string to another string replacing all, part, or none of the string.

Syntax elements	Description
string1	string from which substring is extracted to be inserted into *string2*.
string2	string into which *substring* is inserted; expressed as variable name, dynamic array reference, or literal enclosed in quotes.
start#	position of first group in *string2* to be replaced. If *start#* is specified as 0, it defaults to 1. If *start#* is less than zero, its absolute magnitude is used. If *start#* is greater than number of groups in *string2*, as many null groups as necessary are added and replacement groups are appended to *string2*.
replace#	number of groups in *string2* to replace with groups from *string1*, according to the following rules, where r is replace#: *r* > 0 r groups of *string2* are replaced by the first r groups in *string1*. If *r* is greater than number of groups in *string1*, replacement stops when *string1*is exhausted. *r* = 0 All *string1* inserted before *start#* group in *string2*.

Syntax elements	Description
	r < 0 Number of groups specified by absolute magnitude of *r* are deleted from *string2*, starting with *start#* group. All *string1* inserted at this position (unless *start#* is also less than zero, in which case nothing is inserted).
delim-char	character to delimit groups in both strings. Default is an attribute mark. If more than one character is specified, only the first is used.

HEADING *exp*

Pages current output device and prints text at top of page. The following special control characters can be used in HEADING and FOOTING statements:

Syntax elements	Description
'C{n}'	Centre line (in field of n characters)
'D' or \\	Current date
'T' or \	Current time and date
'L' or]	Return and linefeed
'P' or ^	Current page number
'PP' or ^^	Current page number right justified in 4 spaces
'N'	Inhibit paging
' '	Two quotes print a single quote

3.10 I

ICONV(*expr*,*conv*)

Performs English input conversions.

Syntax elements	Description
expr	expression to convert (not including system delimiters).
conv	input conversion, specified as string in quotes, can be: D, MC, MD, ML, MP, MR, MT, MX or T (see Appendix C).

IF *expr* **THEN** *stmnt*{;*stmnt* }...**ELSE**
stmnt

.

.

.

END

or

IF *expr* **THEN**

stmnt

.

.

.

END {**ELSE** *stmnt* {;*stmnt* }...}

or

IF *expr* **THEN**

stmnt

.

.

.

END {**ELSE**

stmnt

.

.

.

END}

or

IF *expr* **ELSE**

stmnt

.

.

.

END

Allows conditional execution of statement sequence.

Syntax elements	Description
expr	any arithmetic, string, logical, or pattern-matching expression.

{**$**}**INCLUDE** *item* {**FROM** *filespec*}

or

{**$**}**INCLUDE** *filespec item*

Stores large or commonly used sections of code, such as **COMMON** or **EQUATE** areas, outside source code item. Cannot be used with source compacted programs.

Syntax elements	Description
item	item-id of source code item to include.
filespec	file containing item to include. If omitted, item is retrieved from file containing item being processed.

INDEX(*string*,*substring*,*substr-occur*)

Returns starting column number of substring in string.

Syntax elements	Description
substr-occur	integer giving the occurrence of substring to use.

INPUT *var*{,*length*}{:}{_} {**WITH** *delim-mask*} {FOR *time* [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]}

or

INPUT **@**(*c*,*r*) {:} *var* {,*length*}{:}{_} {*format-mask*} {**WITH** *delim-mask*} {**FOR** *time* [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]}

INPUT prompts for input from terminal or **DATA** stack. **INPUT@** prompts for input from terminal at specified cursor position while displaying existing var contents; also provides format masking and pattern matching. See also **INPUTERROR**, **INPUTNULL** and **INPUTTRAP**.

Syntax elements	Description
var	variable or array to which input data is assigned.
length	maximum number of characters to be entered, after which an automatic RETURN is executed. If *length* is not specified, maximum input is 240 characters.
:	inhibits output of RETURN/LINEFEED. Cursor stays positioned after input.
_	valid only with length. When that number of characters has been input, program waits for a RETURN. Attempted entry of other characters sounds terminal 'bell'.
delim-mask	mask (up to 256 characters) defining input delimiters. If used, overrides default delimiter of RETURN. To retain RETURN as an input delimiter, include it in delim-mask.
time	timeout on input, specified in tenths of seconds as an integer between 0 and 32,767. If 0 or negative, result is no timeout. If input is entered within time, THEN clause executed; otherwise, ELSE clause executed.
c,r	cursor column and row position where user prompted.

Syntax elements	Description
:	optionally used after (c,r) of INPUT@; has no significance except compatibility with PRINT @ statement.
format-mask	standard format string. If input data is consistent with this it is formatted and displayed at cursor position. If not (too long or non-numeric where numeric required) an error is displayed.

INPUTERR{**OR**} {*message*}

Prints prompt message on terminal status line or last line. If string omitted, clears terminal status line.

If used in a PRE-WRITE or PRE-DELETE file trigger, aborts the file operation.

Syntax elements	Description
message	literal in quotes, variable or dynamic array reference.

INPUTNULL {*char*}

Defines the single character that, in response to an **INPUT@** statement, causes null value to be assigned to variable. Default is underscore (_).

Note
Entering just RETURN at **INPUT@** prompt leaves existing variable alone.

INPUTTRAP *char-list* [**GO**{**TO**} | **GOSUB**] *label-list*

Defines label to branch to according to single trap character entered at next INPUT@ statement.

Syntax elements	Description
char-list	string consisting of single characters to be compared with character entered at INPUT@ statement.
label-list	names of labels, separated by commas. Number of labels must equal number of characters in char-list.

INS *string* **BEFORE**

dyn-array<attr#{,val#{,subval#}}>

Inserts attribute, value or subvalue into dynamic array. (This statement supersedes the **INSERT** function.)

Syntax elements	Description
string	value to insert into the dynamic array: it may be a dynamic array reference itself.
attr#	attribute position within referenced dynamic array.

Syntax elements	Description
val#	value position within referenced attribute.
subval#	subvalue position within referenced value.

INSERT(*dyn-array*,*attr#*,*val#*,*subval#*,*string*)

or

INSERT(*dyn-array*,*attr#* {,*val#* };*string*)

Inserts attribute, value or subvalue into dynamic array. (INSERT function has been replaced by INS statement but is maintained for compatibility).

Syntax elements	Description
string	value to insert: variable or literal in quotes. May not contain system delimiters.

Note

If *attr#*, *value#* or *subval#* contains -1, string is inserted after last attribute, value or subvalue indicated. Otherwise, string is inserted before specified attribute, value or subvalue.

INT(*expr*)

Returns integer value of given expression.

3.11 L

LEN(*string*)

Returns numeric value of length of string.

{**LET**} *var* = *expr*

Assigns value to variable.

Syntax elements	Description
var	variable where result of expression will be stored.

LN(*expr*)

Calculates logarithms to base 'e'.

LOCATE *expr* **IN** *dyn-array*{<*attr#* {,*val#*}>}{,*start#*} {**BY** *sequence*} **SETTING** *setting-var* [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

or

LOCATE(*expr*,*dyn-array*{,*attr#*{,*val#*{,*start#*}}}; *setting-var* {;*sequence*}) [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Finds position of expression within dynamic array or within attribute or value of a dynamic array.

Syntax elements	Description
expr	string or value being searched for expressed as variable name, numeric constant, literal in quotes or function.
attr#	attribute number within dynamic array being searched.
val#	value number within dynamic array being searched.
start#	attribute or value where search begins. Default is 1.
setting-var	variable to be assigned position of search expression.
sequence	specifies that values are to be sorted as follows (enclose parameter shown in quotes): **AL** Ascending, left justified. **AR** Ascending, right justified. **DL** Descending, left justified. **DR** Descending, right justified. **AR** and **DR** are valid for numeric values only.

LOCK *lock-val* {**THEN** *stmnt*(*s*)} {**ELSE** *stmnt*(*s*)}

Sets execution lock, so another program cannot set the same lock until program that set the lock unlocks it or exits. Halts if lock already set and no ELSE clause.

Syntax elements	Description
lock-val	execution lock to be set. DataBasic and Proc share 256 execution locks, numbered 0 to 255.

LOOP {**VARYING** *var* = *startval* {**STEP** *inc* }}

{ *stmnt*(*s*) }

{

{[**WHILE** | **UNTIL**] *limit* {**DO**}

{ *stmnt*(*s*) }}

}...

REPEAT

Constructs program loops with optional counter var.

Syntax elements	Description
Var	variable for counting iterations through loop.

Syntax elements	Description
Startval	expression, value used as start counter value in *var*.
inc	number by which to increment *var*. Default is 1. *inc* may be negative, causing the loop to count down.
limit	expression that evaluates to true (1) or false (0). Can be **READNEXT**, **READPREV** or **LOCATE** statement.

3.12 M

MAT *array* = *expr* or **MAT** *array1* = **MAT** *array2*

The first form assigns the same value to every element in an array. The second form copies each element of array2 to corresponding element of array1. The number of elements in each must be equal.

Syntax elements	Description
array, array1, array2	any dimensioned array..
expr	value expressed as variable name, literal in quotes, numeric constant, or function.

MATBUILD *var* **FROM** *array*{,*start*{,*end* }} {**USING** *delim-char*}

Builds string variable from dimensioned array (inverse of **MATPARSE**).

Syntax elements	Description
var	destination variable for data built from array.
start, end	positions from which to retrieve array elements.
delim-char	optional single character to insert between elements.

MATPARSE *array*{,*start*{,*end* }} **FROM** *string-var* {**USING** *delim-char*} {**SETTING** *elements-var*}

Assigns elements of string variable to variables of dimensioned array (inverse of **MATBUILD**).

Syntax elements	Description
start, end	positions from which to assign array elements.
string-var	source variable for data to assign to array elements.
delim-char	delimiter of elements of *string-var*. Value from X'00' to X'FE' enclosed in quotes. Default is attribute mark.
elements-var	variable to which is assigned number of elements of array that are assigned a value from *string-var*.

MATREAD *array* **FROM** {*file-var*,}*item-id* {**SETTING** *setting-var*} {**LOCKED** *stmnt*(*s*)} [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Reads file item and assigns each attribute to consecutive elements of dimensioned array (vector).

MATREADU *array* **FROM** {*file-var*,}*item-id* {**SETTING** *setting-var*} {**LOCKED** *stmnt*(*s*)} [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Locks item, then reads it and assigns each attribute to consecutive elements of dimensioned array (vector).

MATWRITE *array* **ON** {*file-var*,} *item-id* {**SETTING** *setting-var*} {**ON ERROR** *stmnt*(*s*)}

Writes dimensioned array (must be vector) to file item. Releases any item lock previously set.

MATWRITEU *array* **ON** {*file-var*,} *item-id* {**ON ERROR** *stmnt*(*s*)}

Writes dimensioned array (must be vector) to file item. Leaves previously locked item locked.

MAXIMUM(*dyn-array*)

Returns maximum numeric element in dynamic array (non-numeric values are ignored). Null is returned if all elements are non-numeric and non-null. Null elements are evaluated as zeros.

MINIMUM(*dyn-array*)

Returns minimum numeric element in dynamic array (non-numeric values are ignored). Null is returned if all elements are non-numeric and non-null. Null elements are evaluated as zeros.

MOD(*expr1*,*expr2*)

Calculates modulo of two expressions.

Syntax elements	Description
expr1, expr2	any valid expressions, strings, substrings, or values.

3.13 N

NOT(*expr*)

Returns logical inverse of expression.

NULL

Specifies no operation.

NUM(*expr*)

Determines whether *expr* is numeric. Returns 1 if *expr* number or numeric string, else returns 0.

3.14 O

OCONV(*expr*,*conv*)

Performs English output conversions.

Syntax elements	Description
expr	expression to convert (not including system delimiters).
conv	output conversion, specified as string in quotes, can be: D, G, MC, MD, ML, MP, MR, MT, MX or T (see Appendix C).

ON *expr* **GOSUB** *stmnt-lbl*{,*stmnt-lbl*...}

or

ON *expr* **GO**{***TO***} *stmnt-lbl*{,*stmnt-lbl*...}

Transfers control to internal subroutine or label determined by current value of given expression.

Syntax elements	Description
expr	any DataBasic expression that evaluates to integer.

OPEN *filespec* {**TO** *file-var*} {**SETTING** *setting-var*} [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Selects file for subsequent read, write or other update.

Syntax elements	Description
filespec	specified as literal (in quotes) or variable.
file-var	variable to which file-name is assigned. If TO *file-var* is specified, file is assigned to it for subsequent reference and can be passed to other programs. If omitted, file is assigned to an internal default file variable that is used by subsequent statements not specifying a file variable.

3.15 P

PAGE {*page*}

Advances current output device to next page and prints heading or footing. Page is an expression giving page number to reset on next page (default is to leave sequence unchanged).

PERFORM *TCL-string* {**PASSLIST** {*sel-var1*}} {**RTNLIST** {*sel-var2*}} {**CAPTURING** *capture-var*} {**SETTING** *setting-var*} {**PASSDATA** *expr*} {**RTNDATA** *rtn-var*}

Executes TCL commands. DATA statement can be used to stack further input. PERFORM can be nested to 8 levels. Clauses can be in any order. A SELECT 'performed' with no RTNLIST clause creates a 'pending' list which is passed to the next PERFORM if it has no PASSLIST clause. PASSLIST, RTNLIST, PASSDATA and RTNDATA clauses are not supported when executing a SYS command.

Syntax elements	Description
TCL-string	valid TCL command (variable or literal in quotes).
PASSLIST *sel-var1*	passes Select list to the called processor. If sel-var1omitted, default list variable is passed (see GETLIST and SELECT statements and RTNLIST clause).
RTNLIST *sel-var2*	returns Select list from called processor. If sel-var2 omitted, list replaces default list variable used in READNEXT or PERFORM, PASSLIST clause.
CAPTURING *capture-var*	captures text otherwise displayed. Each output line becomes an attribute. Printer output is not captured.
SETTING *setting-var*	assigns error messages and their parameters to a variable. Each message is returned as a separate attribute, with parameters separated by value marks. First value is the message number.
PASSDATA *expression*	passes data to COLLECTDATA statement in called program.
RTNDATA *rtn-var*	retrieves data returned by RTNDATA statement in called program.

POSITION *index-var* [= *location* | **END**] {**SETTING** *setting-var*} [**THEN** *stmts*(*s*) | **ELSE** *stmt*(*s*)]

Positions pointer to first index element with key equal to or greater than value location, or to last item in index if END specified.

PRECISION *prec-val*

Sets degree of precision to which values are calculated in multiply, divide, SQRT and runtime conversion of strings to numbers (otherwise default 4 is used). Functions SIN, COS, TAN, PWR, LN calculate to a fixed precision of 5. Addition and subtraction give a result with the precision of the operand with greater precision. Programs can include multiple PRECISION statements.

Syntax elements	Description
prec-val	number (from 0 to 99) of decimal places to which values are calculated and truncate.

PRINT {**ON** *print-val* } {*print-list* }

Outputs data to device selected by PRINTER statement (by default, to terminal).

Syntax elements	Description
print-val	print report number (from 1 to 127).
print-list	single expression or a series of expressions, separated by commas or colons. Expressions can be text strings in quotes, variables that evaluate to text strings, or expressions that denote output formatting (including format strings). See also @ function.

PRINTER [**ON** \| **OFF** \| **CLOSE**]

PRINTER ON directs subsequent program output to printer (current spooler assignment); **PRINTER OFF** directs output to terminal; **PRINTER CLOSE** prints all data currently stored in spooler immediately.

PRINTERR *error-expr* {**FROM** *file-var*}

Prints error messages from ERRMSG or from *file-var*.

Syntax elements	Description
error-expr	evaluates to item-id of error message. Can be literal (in quotes) or variable. Can also contain parameters to be printed via A, A(n) or R(n) message format codes in the error message. (Separate parameters from item-id by system delimiter other than segment mark.)
file-var	Identifies a file other than the ERRMSG file.

PROCREAD *var* [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Reads data from Proc primary input buffer (PIB).

Syntax elements	Description
var	variable to which string value of PIB is assigned.

PROCWRITE *string*

Writes data to Proc primary input buffer (PIB).

Syntax elements	Description
string	string value that is written to Proc PIB: expressed as variable or literal in quotes.

PROMPT *prompt-char*

Selects character used to prompt user for input.

Syntax elements	Description
prompt-char	single character given by literal (in quotes) or variable.

PTR (*fclass*,*fsubclass*{,*param*}...)

Generates printer control string interpreted by despooler according to printer definition assigned.

PWR(*expr1*,*expr2*)

or

PWR expr1^expr2

Calculates variable raised to a power.

Syntax elements	Description
expr1, expr2	expressions that evaluate to numbers. If *expr1* negative, *expr2* must be integer.

3.16 R

READ *dyn-array* **FROM** {*file-var*,}*item-id* {**SETTING** *setting-var*} {**LOCKED** *stmnt*(s)} [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Reads file item and assigns its value, as dynamic array, to a variable. If a LOCKED clause is included, a successful read locks the item read (as if the statement were a READU).

READLIST *dyn-array* **FROM** *list-id* {*account*} {**SETTING** *setting-var*} [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Reads list-id from POINTER-FILE and assigns it to a variable as a dynamic array for program manipulation.

READNEXT *var*{,*vmc-var*{,*svmc-var*}} {**FROM** [*select-var* | *list-var* | *index-var*] {**RETURNING** *key-var*}} {**SETTING** *setting-var*} [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Reads next item-id from select list, list variable or index. If FROM clause omitted, default *select-var* is used.

Syntax elements	Description
var	variable to which string value of each next item-id is assigned.
vmc-var	variable assigned a value count for position in an attribute when an exploding sort has been done using BY-EXP or BY-EXP-DSND in an English command.
svmc-var	variable assigned a subvalue count for position in a multivalue when an exploding sort has been done using BY-EXP-SUB or BY-EXP-SUB-DSND in an English command.

Syntax elements	Description
select-var	select list from which item-id is read.
list-var	name of a list variable.
index-var	name of a variable identifying an index.
key-var	variable to which key value of index item is assigned.

READPREV *var* {,*vmc-var*{,*svmc-var*}} **FROM** *index-var* {**RETURNING** *key-var*} {**SETTING** *setting-var*} [**THEN** *stmt*(*s*) | **ELSE** *stmnt*(*s*)]

Reads previous sequential item-id from index. Parameters are as defined for **READNEXT**.

READT *var* [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Reads next record from magnetic tape unit.

Syntax elements	Description
var	variable to which next record is assigned.

READU *dyn-array* **FROM** {*file-var*,}*item-id* {**SETTING** *setting-var*} {**LOCKED** *stmnt*(*s*)} [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Locks file item, then reads it and assigns its value, as dynamic array, to a variable.

READV *var* **FROM** {*file-var*,}*item-id*, *attr#* {**SETTING** *setting-var*} {**LOCKED** *stmnt*(*s*)} [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Reads attribute value from item and assigns its string value to specified variable.

READVU *var* **FROM** {*file-var*,} *item-id*, *attr#* {**SETTING** *setting-var*} {**LOCKED** *stmnt*(*s*)} [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Locks file item, then reads attribute value from item and assigns its string value to specified variable.

RECEIVE *data* **FROM** *sess-var* {,*ref* } {**USING** *funct* {,*qualifier*}} {**SETTING** *setting-var*} [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

or

RECWAIT *data* **FROM** *sess-var* {,*ref*} {**USING** *funct* {,*qualifier*}} {**TIMEOUT** *minutes*} {**SETTING** *setting-var* } [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

To receive data sent by communicating program. **RECEIVE** executes **ELSE** clause if data not available. **RECWAIT** waits indefinitely or for minutes if data not available, executes **ELSE** clause if error or timeout.

Syntax elements	Description
data	variable to which received data is assigned.
ref	numeric reference for data.
funct	further numeric reference.
qualifier	string qualifier.

RELEASE {{*file-var*,} *item-id* } {**SETTING** *setting-var*}

Unlocks items that have been locked for update. If *file-var* not specified, file most recently opened without *file-var* is unlocked. If neither *file-var* nor item-id are specified, all items locked by program are unlocked.

[**REM** | ***** | **!**]

Marks comments which do not affect program execution: to insert comment, type **REM**, * or ! at beginning of statement, followed by text.

REM(*expr1*,*expr2*)

As synonym function **MOD**.

REMOVE *var* **FROM** *dyn-array* **SETTING** *setting-var*

Successively extracts elements from dynamic array without altering its contents. Leaves that array's 'remove' pointer at next element (to reset pointer to start, assign array to itself).

Syntax elements	Description
var	variable to which substring is assigned.
setting-var	variable to which code is assigned corresponding to system delimiter encountered: 0 = end of array 4 = SVM (252) 1 = SM (255) 5 = SB (251) 2 = AM (254) 6 = (250) 3 = VM (253) 7 = (249)

REPLACE(*dyn-array*,*attr#*,*val#*,*subval#*,*string*)

or

REPLACE(*dyn-array*,*attr#* {,*val#* };*string*)

Replaces attribute, value or subvalue in dynamic array. (Now replaced by direct dynamic array referencing).

Syntax elements	Description
string	replacement value. It may not contain any system delimiters. If *attr#*, *value#* or *subval#* contain a -1, string is inserted after the last attribute, value or subvalue indicated. Otherwise, string is inserted before specified attribute, value or subvalue.

RETURN {**TO** *label*}

Transfers control from subroutine to line after **GOSUB** or **CALL** that called it, or to label.

REWIND [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Rewinds magnetic tape unit to BOT. If tape unit has not been attached, **ELSE** clause is executed.

RND(*expr*)

Returns random number between zero and *expr-1* inclusive (absolute value of *expr* is used).

ROUND (*x*,*y*)

Rounds numeric x to nearest y decimal places.

[**RQM** | **SLEEP**] {*wake-val* }

Terminates program's current timeslice, and causes program to sleep for, or until, specified time.

Syntax elements	Description
wake-val	Either seconds to sleep (integer or fraction), or wakeup time given in quotes in 24-hour format. Default is 1 second.

RTNDATA *expr*

Returns data to **RTNDATA** clause of **PERFORM** statement in program that executed this program.

3.17 S

SELECT *variable* {**TO** *list-var*} {**SETTING** *setting-var*}

or

SELECTE {**TO** *list-var*} {**SETTING** *setting-var*}

or

SELECT {*file-var*} {**TO** *select-var*} {**SETTING** *setting-var*}

or

SELECT *file-var,index-name* **TO** *index-var* {**SETTING** *setting-var*}

Builds item list from variable elements (to subvalue level), or (**SELECTE**) from list generated externally via TCL, or sets up pointer to file or index. List-var can be used by **READNEXT** and **PERFORM** statements. Select-var can be used by **READNEXT** only. Index-var can be used by **POSITION**, **READNEXT** and **READPREV**. If TO clause omitted, default select/list variable is set up for use by appropriate statements. See **READNEXT** for parameters.

SEND *data* {**TO** *sess-var*{,*ref* }} {**USING** *funct*{,*qualifier*}} {**SETTING** *setting-var*} [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Sends data to communicating program.

Syntax elements	Description
data	expression giving data to send.
ref	expression giving numeric reference for *data*.
funct	expression giving further numeric reference.
qualifier	expression giving string qualifier for *data*.

SENTENCE()

Returns the last Proc, PERFORM or TCL statement used.

SEQ(*expr*{,*size*})

Converts ASCII character to its corresponding numeric value where size is from 1 to 4 bytes (default 1).

SIN(*expr*)

Calculates sine of an angle.

Syntax elements	Description
expr	expression giving angle in degrees: 2 Pi radians = 360 degrees.

SLEEP {*wake-val* }

See synonym statement **RQM**.

SOUNDEX (*expr*)

Converts string to its phonetic equivalent.

SPACE(*sp-val*)

Generates string of space characters.

Syntax elements	Description
sp-val	number of blank spaces.

SPOOLER(*sp-funct*{,*line#*/*account*})

Returns spooler status information.

Syntax elements	Description
sp-funct	number that determines spooler function to return: 1 Returns SP-STATUS information. 2 Returns SP-JOBS and password information. 3 Returns SP-ASSIGN information for current port. 4 Returns job information for current port.
line#/account	line number or account depending on sp-funct. Account if sp-funct=2. Line number if sp-funct=3.

SQRT(*expr*)

Calculates square root of expression.

Syntax elements	Description
expr	expression greater than or equal to zero.

SQUOTE(*expr*)

Returns specified string enclosed in single quotes.

Syntax elements	Description
string-var	field of characters expressed as variable.

STOP {*msg-id* {,*msg-expr*}...}

Halts execution of program and optionally displays message.

Syntax elements	Description
msg-id	numeric item-id of item in the ERRMSG file containing message. Message is printed when **STOP** is executed
msg-expr	expressions to be printed as part of message. These are processed on first-in first-out basis and are printed via message format codes **A**, **A**(*n*) and **R**(*n*).

STR("*string*",*number*)

Generates string value containing *string* repeated *number* times.

SUB{**ROUTINE**} *ctlg-id* {(*argument-list*)}

Identifies program as external subroutine called by another program. It must be first statement in the program. Precision in calling program and subroutine need not match.

Syntax elements	Description
ctlg-id	name under which this program was cataloged.
argument-list	one or more variable(s), separated by commas, to be assigned values passed via **CALL** statement.

SUMMATION(*dyn-array*)

Returns sum of all numeric elements of dynamic array.

SYSTEM(*sys-element*) = *value*

Allows the states of various system elements to be changed.

Syntax elements	Description
data-element	Number corresponding to system element; valid values are: 2, 3, 5, 7, 30, 35, 37, 38 or 39 (see **SYSTEM** function).

SYSTEM(*sys-element*)

Returns current state of database parameters. Some of these can be assigned values: see **SYSTEM** statement and **ASSIGN**.

Syntax elements	Description
sys-element	number corresponding to parameter to reference (those omitted are not currently used): 0 Returns error message number. 1 Returns 1 if PRINT destination is currently printer. 2 Returns page width. 3 Returns page length. 4 If HEADING statement used, returns number of lines of current page still to print. 5 If HEADING statement used, returns page number. 6 If HEADING statement used, returns line number. 7 Returns terminal type. 9 Returns CPU millisecond count for the calling process, accurate to nearest 20 ms. 10 Returns 1 if stacked input is currently available. 11 Returns 1 if an external list (generated by TCL command SELECT or equivalent) is active. 12 Returns system time in 1/10 second format, accurate to nearest second. 14 Returns 1 if typeahead available, 0 if not. 15 Returns options used with last TCL command

Syntax elements	Description
	used, with up to three attributes: • String of letters used as options. • First numeric parameter. • Second numeric parameter. 16 Returns current level of nesting of PERFORM statement. 18 Returns port number. 19 Returns account name. 20 Returns 1 if program running is cataloged. 21 Returns code for video characteristics supported: 0 Invalid. 1 Video characteristics not supported. 2 Video character requires CRT position. 3 CRT position not required. 22 Returns system configuration as dynamic array with the following attributes: 1 System serial number. 2 Set to 0 (proprietary release 7.x or earlier returns firmware type). 3 Set to 0 (proprietary release 7.x or earlier returns firmware version). 4 1 if Wordmate allowed, 0 if not. 5 Set to 2047 (number of ABS frames for proprietary release 7.x or earlier). 6 Maximum number of active processes. 7 Set to -1 (returns Session Manager's process number on proprietary release 7.x or earlier). 8 Set to 0 (returns maximum FID on proprietary release 7.x or earlier). 9 Number of workspace frames. 10 Maximum process number. 11 1 if UK system, 0 if not. 12 Memory size in Kbytes. 53 Returns zero. 54 Returns zero. 55 Returns system time in milliseconds, accurate to nearest second. 56 Returns 0 (proprietary release 7.x or earlier returns disk data). 57 Returns snapshot of workspace overflow table. 58 Returns spooler assignment data (like SP-LOOK). 60 Returns TCL input statement without verb, options and redundant spaces, and with attribute marks in place of remaining spaces. 61 Returns name of physical account (D-pointer) logged-on to.

Syntax elements	Description
	62 Returns the last input delimiter used. 63 Returns current security setting for SQL Stored Procedures. 64 Returns current security setting for Remote Basic. 65 Returns current security setting for Dictionary Basic. 66 Returns current security setting for User Exits. 67 Returns the item-id of the current security profile. 70 Returns the underlying system, as follows: 0 - Series 19 1 - UNIX 2 – Windows NT/2000 71 Returns the current live version of Reality. 72 Returns the maximum number of tape devices defined on the database. 73 Returns the magnetic tape assignment information. 74 Returns 1 if DDA session level messaging is supported; 0 otherwise.

3.18 T

TAN(*expr*)

Calculates tangent of an angle.

Syntax elements	Description
expr	expression giving angle in degrees: 2 Pi radians = 360 degrees.

TIME()

Returns current time in internal format.

TIMEDATE()

Returns current time and date in external format.

TRANSABORT {*trans-info*} [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Aborts current transaction, undoes any updates to database performed by transaction, and release item locks set during transaction.

TRANSQUERY()

Returns 1 (true) if process executing function is within transaction boundary or 0 (false) if not.

TRANSEND {*trans-info*} [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Denotes end of transaction. Once executed, rollback of transaction's updates is prevented. Item locks set during the transaction are released.

TRANSTART {*trans-info*} {**SETTING** *setting-var*} [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Denotes start of transaction. Any updates after TRANSTART are rolled-back if followed by a TRANSABORT. Item lock release is suspended until TRANSEND or TRANSABORT.

TRIM(*string*{,*remove-char*{,*type*}})

Deletes specified character (or blanks) from *string*.

Syntax elements	Description
remove-char	character to delete, instead of default blank.
Type	one of the following, enclosed in quotes: L Removes all leading remove-char s. T Removes all trailing remove-char s. B Removes leading and trailing remove-char s. A Removes all remove-char s. R Removes redundant remove-char s (default).

TRUNC (*x*,*y*)

Truncates numeric *x* to *y* decimal places.

3.19 U

UNASSIGNED(*var*)

Returns 0 if a value is currently assigned to a variable, otherwise returns 1.

Syntax elements	Description
var	single variable reference only.

UNLOCK {*lock-val* }

Resets execution locks.

Syntax elements	Description
lock-val	execution lock to reset. Default is all execution locks previously set by program. DataBasic and Proc processors use the same execution locks: 0 to 255.

UPCASE(*expr*)

Returns expr with all lower case letters converted to upper case (like conversion MCU).

3.20 V

VARTYPE(*var*)

Returns the type of the variable, as determined at compilation, as a string:

type ordinal {*rows* {*columns*}}, where:

type can be:

V	simple variable
D	dimensioned variable
U	undefined variable

ordinal identifies the group to which it belongs:

0	local variable
1	common variable
2	labelled common block

rows and columns are integers showing the rows and columns belonging to a dimensioned variable.

VARVAL(*var*)

Returns the current value of the variable. This can be <unassigned> or file.variable filename string.

VARVALSET *var* **TO** *expr* [**THEN** *stmnt*(*s*)|**ELSE** *stmnt*(*s*)]

Sets var to the value given by expr. ELSE clause is taken if the variable cannot be updated for any reason.

VARVALTYPE(*variable*)

Returns the current variable type as follows:

00 Cleared, unassigned

01 Scaled binary number

02 Short string

04 File variable

81 String number

82 Indirect string

U Undefined variable

3.21 W

WEOF [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Writes End-Of-File mark (EOF) to tape. If tape unit has not been attached, **ELSE** clause is taken.

WRITE *string* **ON** {*file-var*,}*item-id* {**SETTING** *setting-var*} {**ON ERROR** *stmnt*(*s*)}

Writes variable, as dynamic array, to file item, releasing any lock currently set on the item (unless within a transaction). Compare **WRITEU**.

WRITELIST *string* **ON** *list-id*

Writes string to POINTER-FILE as saved list.

Syntax elements	Description
string	item-ids to save; may be literal list or variable.
list-id	name of list in POINTER-FILE where list will be saved.

WRITET *string* [**THEN** *stmnt*(*s*) | **ELSE** *stmnt*(*s*)]

Writes record to tape.

Syntax elements	Description
string	variable, literal or result of DataBasic expression.

WRITEU *string* **ON** {*file-var*,}*item-id* {**SETTING** *setting-var*} {**ON ERROR** *stmnt*(*s*)}

Writes variable, as dynamic array, to file item. Leaves previously locked item locked after the write.

WRITEV *string* **ON** {*file-var*,}*item-id*,*attr#* {**SETTING** *setting-var*} {**ON ERROR** *stmnt*(*s*)}

Updates attribute value in file item.

Syntax elements	Description
attr#	number of attribute where string is written.

WRITEVU *string* **ON** {*file-var*,}*item-id*,*attr#* {**SETTING** *setting-var*} {**ON ERROR** *stmnt*(*s*)}

Updates attribute value in file item, and leaves previously locked item locked after the write.

Syntax elements	Description
attr#	number of attribute where expression is written.

3.22 X

XTD(*expr*)

Converts a hexadecimal value to decimal.

Section 4: Related TCL Commands

DataBasic-related TCL Commands

For more details of the following, see TCL command descriptions:

Command	Description
BASIC	Compiles a DataBasic program.
BLIST	Lists source code with logical indenting.
BREF	Produces sorted cross-reference list of variables and labels.
BVERIFY	Verifies object code of cataloged program (SYSMAN/SYSPROG only)
CATALOG	Catalogs DataBasic program (creates MD entry and shared executable item).
CLEAR-BASIC-LOCKS	Resets execution locks (SYSMAN/SYSPROG only).
CONVERT.OBJECT	Converts object code items to current format. Do not run from TCL.
DB	Provides prompts for edit, compile, catalog, run and debug of a program.
DEBUG	Runs program under debugger control.
DECAT	Loads object code item into source file.
DELETE-CATALOG	Deletes POINTER-FILE and MD entries for cataloged program.
ECOPY	Expands and copies compressed items (obsolescent).
LISTPF	Lists POINTER-FILE (SYSMAN/SYSPROG only).
LOAD-BNF	Loads alternative compilers.
PRINT-CATALOG	Prints time and date cataloged programs were compiled.
RUN	Executes compiled DataBasic program.
UPGRADE.BASIC.OBJECT	Converts POINTER-FILE items to current format.
VERIFY-SYSTEM	Verifies integrity of system DataBasic programs (SYSMAN/SYSPROG only).

Section 5: Debugger

5.1 $

[**£** | **$**]

Displays number of program line to be executed.

5.2 /

{**X**} / [*var* | *]

Displays and allows you to modify value of a variable.

Syntax elements	Description
X	for display and input in hex (default is format chosen by last DX command).
var	name of simple variable, array or array element.
*	displays all variables and arrays in program.

5.3 ?

?

Displays name of currently executing program.

5.4 @

@

Inhibits a break if **DEBUG** statement is encountered; it toggles function of **DEBUG** statement.

5.5 A

A

Displays number of program line to be executed. (synonym of $.)

5.6 B

B*var operator operand* {**&***var operator operand* }...

Adds entry to Breakpoint Table. Conditions can be ANDed (using **&**). Each condition must then be met for break to occur. SYS2 privileges required.

Syntax elements	Description
var	simple variable, array, array element, or $ character to reference next line to be executed.
operator	any of the following logical operators: =, #, <, >, <= or >=

Syntax elements	Description
operand	variable, array element, string, literal in single or double quotes, numeric literal or literal @.

5.7 D

D

Displays Break and Trace tables. SYS2 privileges required.

DE{**BUG**} or **Debug**

Exits from DataBasic Debugger to rdb debug.

5.8 E

E{*n*}

Creates a break in program execution after defined number of instructions. SYS2 privileges required.

Syntax elements	Description
n	cardinal number defining number of instruction lines that will be executed before a break occurs.

END [**RETURN** |**LINEFEED**]

Terminates DataBasic program and exits Debugger.

RETURN returns control to TCL; LINEFEED returns control to next statement in Proc or after PERFORM statement.

5.9 G

G{line-num} or **LINEFEED**

Resumes normal execution of DataBasic program until next execution break is encountered.

Syntax elements	Description
line-num	line-number where program execution continues.

5.10 K

K{*n*}

Kills one or all of breakpoint conditions in breakpoint table. SYS2 privileges required.

Syntax elements	Description
N	cardinal number in range 1 to number of breakpoint sets. Specified breakpoint is deleted and other breakpoints remain unchanged. Default is all conditions.

5.11 L

L {*n*{-*m*}}

or

L {*n*{,*m*}}

or

L*

or

W {*n*}

Lists source code lines from executing program. **L** lists line about to be executed. **L***n* lists line *n*. **L***n*-*m* lists from line *n* to line *m*. *Ln*,*m* lists m lines starting at line *n*. **L*** lists all lines. **W** lists page up to and including line about to be executed. **W***n* lists page up to and including line *n*.

LDT or **LDP** or **LDB**

Sends debugger output to terminal, printer or both (respectively).

LP

Toggles output from PRINT statements between terminal and spooler.

5.12 N

N{*n*}

Bypasses *n* breakpoints. SYS2 privileges required.

5.13 O

OFF

Terminates program and logs you off database.

5.14 P

P

Suppresses all output from program to terminal, so that only output from debugger is displayed. **P** toggles status.

PC

Forces printing of any data waiting to be output.

PR{*opts*}

Toggles Profiler: when on, it increments a counter for each program line every time that line is executed. SYS2 privileges required.

Syntax elements	Description
P	Sends Profiler information to printer.
T	Sends Profiler information to terminal.

5.15 S

S

Displays internal or external subroutine return stack. SYS2 privileges required.

5.16 T

T {*var*}

Switches display of trace table to character format. SYS2 privileges required.

Syntax elements	Description
var	variable to be traced.

5.17 U

U{*entry-num*}

Deletes variables from trace table. SYS2 privileges required.

Syntax elements	Description
entry-num	number of trace entry in range 1 to the number of trace entries. Default is entire table.

5.18 V

V*m* {,*n*}

Sets the number of source code lines to be displayed automatically at every entry into the debugger. Use **V0** to turn feature off. SYS2 privileges required.

Syntax elements	Description
m	number of lines from current forward.
n	number of lines prior to current line.

5.19 W

W

Lists page of source code item. Refer to **L** command. SYS2 privileges required.

5.20 X

X {*var*|*}

Displays the value of a variable in hex and allows you to change it. Also, displays all variables in hex without ability to change. SYS2 privileges required.

Syntax elements	Description
var	variable to be displayed.

5.21 Z

Z *filespec item-id*

Assigns symbol table to program being debugged. SYS2 privileges required.

Section 6: Appendix A Statements and Intrinsic Functions by Category

6.1 Statements by Category

Accessing Proc		
PROCREAD	PROCWRITE	
Accessing TCL		
CHAIN	PERFORM	
Assignment		
CLEAR	MAT	LET
Branching		
CASE	IF	ON GO{TO}
GO{TO}	INPUTTRAP	
Data Definition		
CLEAR	DIMENSION	PRECISION
COMMON	EQUATE	
Data Input		
DATA	INPUT@	INPUTTRAP
GROUPSTORE	INPUTERR{OR}	PROCREAD
INPUT	INPUTNULL	PROMPT
Data Output		
CRT	INPUTERR{OR}	PRINTER
FOOTING	PAGE	PROCWRITE
HEADING	PRINT	
Dimensioned Arrays		
DIM{ENSION}	MATPARSE	MATWRITE
MAT	MATREAD	MATWRITEU
MATBUILD	MATREADU	

Dynamic Arrays		
DEL	GROUPSTORE	REMOVE
FIND	INS	
FINDSTR	LOCATE	
Execution Locks		
LOCK	UNLOCK	
File I/O		
CLEARFILE	MATWRITE	READVU
CLOSE	MATWRITEU	SELECT{E}
DELETE	OPEN	WRITE
DELETELIST	READ	WRITELIST
GETLIST	READLIST	WRITEU
MATREAD	READU	WRITEV
MATREADU	READV	WRITEVU
Item Lists and Indexes		
GETLIST	READNEXT	SELECT
PERFORM	READLIST	SELECTE
POSITION	READPREV	WRITELIST
Item Locks		
MATREADU	READU	WRITEU
MATWRITE	READVU	WRITEV
MATWRITEU	RELEASE	WRITEVU
READ	WRITE	
Looping		
FOR	LOOP	NEXT

Miscellaneous Control		
$CHAIN	DEBUG	PRINTERR

Miscellaneous Control		
$INCLUDE	ECHO	REM
ASSIGN	INCLUDE	RQM
BREAK	NULL	SLEEP
Program Termination		
ABORT	END	STOP
Program-to-Program Communication		
ACCEPT	DISCONNECT	RECWAIT
CONNECT	RECEIVE	SEND
Subroutine Branching		
CALL	INPUTTRAP	RETURN
GOSUB	ON GOSUB	SUB{ROUTINE}
System Interaction		
ASSIGN	ECHO	RQM
BREAK	ENTER	RTNDATA
CALL	PERFORM	SLEEP
CHAIN	PRINTERR	
COLLECTDATA	PROCREAD	
DATA	PROCWRITE	
DEBUG	PROMPT	
Tape I/O		
READT	WEOF	
REWIND	WRITET	
Transaction Handling		
TRANSABORT	TRANSABORT	TRANSTART

6.2 Intrinsic Functions by Category

Bit Manipulation		
BITCHANGE	BITLOAD	BITSET
BITCHECK	BITRESET	
Format Conversions		
ASCII	EBCDIC	SEQ
CHAR	FMT	UPCASE
DOWNCASE	ICONV	XTD
DTX	OCONV	
Format Conversions		
ALPHA	NOT	NUM
Manipulating Dynamic Array Elements		
DELETE	MAXIMUM	SUMMATION
EXTRACT	MINIMUM	
INSERT	REPLACE	
Maths/numeric		
ABS	MOD	SIN
COS	PWR	SQRT
EXP	REM	TAN
INT	RND	TRUNC
LN	ROUND	
Miscellaneous		
@	GROUP	SYSTEM
BCC	PTR	TRANSQUERY()
CRC	SENTENCE()	UNASSIGNED
DQUOTE	SPOOLER	
GETMSG	SQUOTE	

String/Substring Manipulation		
CHANGE	DECRYPT	SOUNDEX
CHECKSUM	ENCRYPT	SPACE
COL1/COL2	FIELD	STR
CONVERT	FOLD	TRIM
COUNT	INDEX	
DCOUNT	LEN	
Time and Date		
DATE()	TIME()	TIMEDATE()
Variable Checking		
VARTYPE	VARVAL	VARVALTYPE

Section 7: Appendix B Reserved Words

Reserved words cannot be used as variable names, statement labels or subroutine names - error message is 'Bad Statement'. If word is nested in loop, message may refer to start of loop, making debugging difficult.

The following words may not be used as variable names:

AND	AT	BEFORE
BY	CAPTURING	CAT
CHAIN	DO	ELSE
EQ	FOR	FROM
GE	GO	GOSUB
GOTO	GT	IN
INCLUDE	LE	LET
LOCKED	LT	MATCH
MATCHES	NE	ON
OR	PASSDATA	PASSLIST
REMOVE	REPEAT	REPLACE
RETURNING	RTNDATA	RTNLIST
SETTING	STEP	THEN
TIMEOUT	TO	UNTIL
USING	WITH	WHILE

The following words are the names of functions and may not be used as array names

ABS	ALPHA	ASCII
BCC	BITCHANGE	BITCHECK
BITLOAD	BITRESET	BITSET
CHANGE	CHAR	CHECKSUM
COL1	COL2	CONVERT
COS	COUNT	CRC

DATE	DCOUNT	DECRYPT
DELETE	DOWNCASE	DQUOTE
DTX	EBCDIC	ENCRYPT
EXP	EXTRACT	FIELD
FMT	FOLD	GETMSG
GROUP	ICONV	INDEX
INSERT	INT	LEN
LN	MAXIMUM	MINIMUM
MOD	NOT	NUM
OCONV	PWR	PTR
REM	REMOVE	REPLACE
RND	ROUND	SENTENCE
SEQ	SIN	SOUNDEX
SPACE	SPOOLER	SQUOTE
SQRT	STR	SUMMATION
SYSTEM	TAN	TIME
TIMEDATE	TRANSQUERY	TRIM
TRUNC	UNASSIGNED	UPCASE
VARVAL	VARTYPE	VARVALTYPE
XTD		

Section 8: Appendix C Conversions

Conversions and DataBasic

A subset of the conversion codes supported by English dictionaries can be used in DataBasic programs, via OCONV and ICONV functions and format strings.

- OCONV and format strings perform 'output' conversion.
- ICONV performs 'input' conversion (like the processing English does on values in an English sentence before comparison with pre-processed file data).

Conversion codes are fully documented in English reference documentation.

The codes which can be used from DataBasic are:

Codes	Description
D	Converts date to external/internal format.
G	Performs group extraction (not ICONV)
MC	Performs character conversion.
MD	Converts integer to decimal number (or vice versa).
ML	Mask decimal, left justify.
MP	Converts packed decimal number to integer (or vice versa).
MR	Mask decimal, right justify.
MT	Converts time to external/internal format.
MX	Converts ASCII to hexadecimal (or vice versa).
T	Extracts character substring from attribute value.
Tfile	Converts by table/file translation. Inefficient if you need to access several items or attributes.

Section 9: Appendix D Conversions

Code	Bold	Underlined	Blanked	Reversed	Flashing	Dimmed
-128	This is video off.					
-129						DM
-130					FL	
-131					FL	DM
-132				RV		
-133				RV		DM
-134				RV	FL	
-135				RV	FL	DM
-136			BK			
-137			BK			DM
-138			BK		FL	
-139			BK		FL	DM
-140			BK	RV		
-141			BK	RV		DM
-142			BK	RV	FL	
-143			BK	RV	FL	DM
-144		UL				
-145		UL				DM
-146		UL			FL	
-147		UL			FL	DM
-148		UL		RV		
-149		UL		RV		DM

-150		UL		RV	FL	
-151		UL		RV	FL	DM
-152		UL	BK			
-153		UL	BK			DM
-154		UL	BK		FL	
-155		UL	BK		FL	DM
-156		UL	BK	RV		
-157		UL	BK	RV		DM
-158		UL	BK	RV	FL	
-159		UL	BK	RV	FL	DM
-160	BD					
-161	BD					DM
-162	BD				FL	
-163	BD				FL	DM
-164	BD			RV		
-165	BD			RV		DM
-166	BD			RV	FL	
-167	BD			RV	FL	DM
-168	BD		BK			
-169	BD		BK			DM
-170	BD		BK		FL	
-171	BD		BK		FL	DM
-172	BD		BK	RV		
-173	BD		BK	RV		DM
-174	BD		BK	RV	FL	

-175	BD		BK	RV	FL	DM
-176	BD	UL				
-177	BD	UL				DM
-178	BD	UL			FL	
-179	BD	UL			FL	DM
-180	BD	UL		RV		
-181	BD	UL		RV		DM
-182	BD	UL		RV	FL	
-183	BD	UL		RV	FL	DM
-184	BD	UL	BK			
-185	BD	UL	BK			DM
-186	BD	UL	BK		FL	
-187	BD	UL	BK		FL	DM
-188	BD	UL	BK	RV		
-189	BD	UL	BK	RV		DM
-190	BD	UL	BK	RV	FL	
-191	BD	UL	BK	RV	FL	DM

Section 10: Appendix E Decimal, Hex and ASCII Table

Dec	Hex	ASCII	Key(s)/Effect
000	00	NUL	CTRL+@
001	01	SOH	CTRL+A, HOME
002	02	STX	CTRL+B
003	03	ETX	CTRL+C
004	04	EOT	CTRL+D
005	05	ENQ	CTRL+E
006	06	ACK	CTRL+F, →
007	007	BEL	CTRL+G
008	08	BS	CTRL+H, BACKSPACE
009	09	HT	CTRL+I, TAB
010	0A	LF	CTRL+J, LF ↓
011	0B	VT	CTRL+K
012	0C	FF	CTRL+L, CLEAR
013	0D	CR	CTRL+M, RETURN
014	0E	SO	CTRL+N
015	0F	SI	CTRL+O
016	10	DLE	CTRL+P
017	11	DC1	CTRL+Q (XON function)
018	12	DC2	CTRL+R (redisplay line)
019	13	DC3	CTRL+S (XOFF function)
020	14	DC4	CTRL+T
021	15	NAK	CTRL+U, ←

022	16	SYN	CTRL+V
023	FR17	ETB	CTRL+W
024	18	CAN	CTRL+X, (cancel line)
025	19	EM	CTRL+Y, SHIFT+TAB (disconnect)
026	1A	SUB	CTRL+Z, ↑
027	1B	ESC	ESC, CTRL+[
028	1C	FS	
029	1D	GS	
030	1E	RS	
031	1F	US	
032	20	SPACE	SPACE
033	21	!	!
034	22	"	"
035	23	#	#
036	24	$	$
037	25	%	%
038	26	&	&
039	27	'	'
040	28	(	(
041	29	)	)
042	2A	*	*
043	2B	+	+
044	2C	,	,
045	2D	-	-
046	2E	.	.

047	2F	/	/
048	30	0	0
049	31	1	1
050	32	2	2
051	33	3	3
052	34	4	4
053	35	5	5
054	36	6	6
055	37	7	7
056	38	8	8
057	39	9	9
058	3A	:	:
059	3B	;	;
060	3C	<	<
061	3D	=	=
062	3E	>	>
063	3F	?	?
064	40	@	@
065	41	A	A
066	42	B	B
067	43	C	C
068	44	D	D
069	45	E	E
070	46	F	F
071	47	G	G

072	48	H	H
073	49	I	I
074	4A	J	J
075	4B	K	K
076	4C	L	L
077	4D	M	M
078	4E	N	N
079	4F	O	O
080	50	P	P
081	51	Q	Q
082	52	R	R
083	53	S	S
084	54	T	T
085	55	U	U
086	56	V	V
087	57	W	W
088	58	X	X
089	59	Y	Y
090	5A	Z	Z
091	5B	[	[
092	5C	\	\
093	5D	]	]
094	5E	^	^
095	5F	_	_
096	60	`	`

097	61	a	a
098	62	b	b
099	63	c	c
100	64	d	d
101	65	e	e
102	66	f	f
103	67	g	g
104	68	h	h
105	69	i	i
106	6A	j	j
107	6B	k	k
108	6C	l	l
109	6D	m	m
110	6E	n	n
111	6F	o	o
112	70	p	p
113	71	q	q
114	72	r	r
115	73	s	s
116	74	t	t
117	75	u	u
118	76	v	v
119	77	w	w
120	78	x	x
121	79	y	y

122	7A	z	z
123	7B	{	{
124	7C	|	|
125	7D	}	}
126	7E	~	~
127	7F	DEL	DELETE
:			
251	FB	[	(SB: start buffer)
252	FC	\	CTRL+\, (SVM: subvalue mark)
253	FD	]	CTRL+], (VM: value mark)
254	FE	^	CTRL+^, (AM: attribute mark)
255	FF	_	CTRL+_, (SM: segment mark)

Key shown is typical, but may vary with terminal type.

About NEC Software Solutions
NEC
Our customers change lives, so we create software and services that get them better outcomes. By innovating when it matters most, we help to keep people safer, healthier and better connected worldwide.
NECSWS.com
1st Floor, Bizspace, iMex Centre,
575-599 Maxted Rd,
Hemel Hempstead HP2 7DX
+44 (0)1442 768445