Reality v10.0

Proc 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

The Proc processor allows you to store a complex series of commands that can later be executed by a single word command. Any sequence of commands that can be executed from a terminal can be stored in a Proc. No compilation is necessary.

Proc can also be used interactively, prompting the user for input then testing and verifying that data. Additionally, Proc can read and update files.

This guide summarizes the main features of Proc, giving a brief description of all commands.

It will be useful to experienced Reality users, Pick users migrating to Reality and to new users exploring the system.

Note
PQN ('new Proc') and PQ ('old Proc') are two different versions of Proc. PQN has the most extensive capabilities and is generally recommended as the version to use. Both versions can be used on the same database, but PQ Procs can not be called from PQN and vice versa. This guide covers both versions (see Appendix C for differences).

Caution
It is outside the scope of this guide to fully define each command in context. If in doubt, check the detailed description in the *Proc Reference Manual*.

1.2 References

Proc Reference Manual

This gives a detailed description of the Proc programming language.

English Reference Manual

Describes input and output conversions for use with Proc.

1.3 Frequently-Used Parameters

The following definitions apply throughout except where otherwise indicated.

cmd

Valid Proc command.

data-sect

Name of data section (if different from dictionary name). It can be literal name (not enclosed in quotes) or direct or indirect reference to buffer or select register containing required name.

DICT
Specifies dictionary section of file.

file

Name of file for command to act on, or file containing Proc to chain to in case of Proc-to-Proc transfer.

filespec

Defines the file for command to act on, Within Proc it has the syntax:

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

item-id

Name of item for command to act on, or consisting of Proc to chain to (in latter case, if omitted, current primary input buffer parameter defines item-id).

transaction-info

Text, in quotes, describing current transaction; default is file name and item-id.

1.4 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: Buffers, Features and Transaction Handling

Creating Procs

Use the TCL command CREATE-FILE to create a file in which to store Procs (if one does not exist already). Use one of the editors to create an item with the form:

item-id	*procname*
att.1	**PQN**
att.2	*Proc command*
:	
:	

Use as many Proc commands as you like. No END command is needed.

To execute the Proc, use an editor to create a calling entry in your local Master Dictionary (MD) (see below) then just type the name of the Proc at TCL. Use the TCL command **TRACE** to execute the Proc if you need to debug it.

Calling Entries

Procs should not be put in the MD because this lengthens execution time for MD verbs and commands. Procs are usually stored in separate files created for them. A short item to call the Proc is then entered in the MD. This MD calling entry has the general form:

item-id	*procname*
att.1	**PQN**
att.2	*(procfile)*

This executes Proc *procname* in file *procfile*.

Labels

Any Proc command can be preceded by a numeric label that identifies it for branching or looping.

Comment Lines

Any line starting with C or * is a comment and is ignored by Proc, unless a multi-command delimiter \ (see below) appears in it in which case anything after the \ is executed.

Multiple Commands

Multiple commands can generally be placed on the same line, separated by subvalue marks (CTRL+\). However, the following should not be followed by another command on the same line: (), [], B, BO, F, F;, FB, F-CLEAR, F-FREE, F-KLOSE, F-OPEN, F-READ, F-UREAD, F-WRITE, GOSUB, H, IBH, IH, MVA, MVD, O, P, RI, RO, RSUB, RTN, U, X, and continuation lines for a T or L command on a previous line. \

Buffers

Proc moves data from active input buffer to primary or secondary output buffer; primary input buffer is usually the active input buffer. Various commands determine the active buffer: see Active Buffers.

Primary Input Buffer (PIB)

Contains Proc name and optional arguments exactly as entered; it is usually used for terminal input and for storing parameters that control the sequence of Proc execution. Using its parameters as storage locations is similar to DataBasic variable assignment. PIB is set to contain value of **CHAIN** or **PERFORM** expression when Proc is executed from DataBasic. A DataBasic program run from Proc (after which control will return to Proc) can alter contents of PIB via **ProcWRITE**. Input buffers are referenced by percent sign (%).

Secondary Input Buffer (SIB)

Contains data input by user at last **IN** command. Input buffers are referenced by percent sign (**%**).

Primary Output Buffer (POB)

Holds any TCL command to be passed to TCL via **P** command for processing. RETURN is automatically placed at end of TCL command. Any command left in POB on exiting Proc is automatically executed. Output buffers are referenced by hash sign (**#**).

Note

Use Proc transfer to call one Proc from another - never execute via output buffer.

Secondary Output Buffer (SOB)

Contains data used in interactive processes when input is requested. Each line of data entered with **H** command must end with less-than character (<) to represent RETURN; if text exceeds 140 characters, each line of 140 characters or less must be separated by two less-than characters (<<). SOB is same as data stack used by **DATA** statement in DataBasic, and contents can be copied elsewhere. Output buffers are referenced by hash sign (**#**).

If Proc is executed from TCL, SOB is initially cleared. If Proc is executed via DataBasic **CHAIN** or **PERFORM**, SOB will contain data put in stack by **DATA** statement; this is not a good way to pass data as Proc input statements ignore stacked data.

Active Buffers

POB is Active Output Buffer (AOB) and PIB is Active Input Buffer (AIB) at Proc start-up and after **STOFF** (Stack Off) command; SOB becomes AOB after **STON** (Stack On) command. For operations requiring two buffers, switch between buffers.

PQ and PQN Procs

The first line of a Proc is PQ or PQN. PQ Procs can only transfer to other PQ Procs and PQN Procs can only transfer to other PQN Procs.

PQ Procs use blanks as parameter delimiters. PQN Procs use attribute marks which allow parameters which are null, contain embedded blanks or match format of Reality file items. PQN has additional commands which take advantage of this delimiter, eg for buffer referencing. Write new applications using PQN Proc (unless using SMA Reality). See Appendix C for summary of differences in command availablility and functionality in PQ and PQN Proc.

Direct Referencing

In PQN Procs, input buffers are referenced by percent sign (%) and output buffers by hash sign (#). It is only necessary to reference SIB when using IF E, IN or MS commands. File buffers referenced by & and select registers by ! - see below.

References to parameters are made by using ordinal or sequential number of parameter within Proc, or by number of column it starts in. T command (at any time during PQN Proc execution) displays contents of parameters of PIB.

Indirect Referencing

The primary input buffer, active output buffer and file buffers can be referenced indirectly by following the buffer reference symbol (%, # or & respectively) with another buffer reference symbol and a literal. If the value is non-numeric, zero used.

File Buffers (PQN only)

File buffers numbered 1 to 20 contain records read from or written to disk. Buffers 19 and 20 are reserved for system Proc use. Data is divided into attributes, separated by attribute marks.

File buffers are referenced only after they have been cleared or opened. Reference by &fb-num.n, that is ampersand, file buffer number, period and numeric value representing attribute (eg &6.2). Attribute 0 contains the item-id. If you reference an attribute beyond the last existing attribute, the required number of null attributes is created.

Select Registers

Select registers numbered 1 to 10 are used for processing lists of item-ids and other multivalued fields.

Select registers are referenced by exclamation mark (!) and register number. Each time a register is referenced the next value is popped off top of stack for processing. If any value is moved into a register (eg via MV command), entire contents of register are destroyed.

Pointers

There is one buffer pointer for active input buffer which can be moved back and forth without affecting data. The two output buffers have one pointer each, which always points to end of data; if moved backward, data after pointer is truncated. You cannot move a pointer further than the last data item.

Transaction Handling

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', every program should lock and unlock the same items in the same sequence.

A set of updates can be identified as a single transaction using TRANSTART and TRANSEND or aborted/undone by TRANSABORT; TRANSQUERYdisplays transaction status of current port. These are TCL commands (equivalent functions are available within DataBasic, ALL and RPL).

Transaction logging is an optional feature which saves all or selected updates to disk.

Time and Date

To load the current system time or date (in internal format) into the active input buffer at the location pointed to by the buffer pointer, use the command: IH%n:F;T: or IH%n:F;D: respectively.

The value n has no significance but the %n causes the rest of the command to be part of the function, not a literal string.

READV Equivalent

A single attribute from an item can be loaded into the active input buffer with command:

IBH%n:Tfile;X;amc;amc: where file is the name of the file and amc is the attribute mark count.

Concatenation

Values can be concatenated using asterisk (*). For example, %1*%2 concatenates values 1 and 2 in input buffer; #1*" "*#2 concatenates values 1 and 2 in output buffer with a space between them.

Section 3: PQN Commands

3.1 ()

(*filespec* {*item-id* }) {*label*}

( ) terminates current Proc and executes another. Execution starts at a label within the called Proc (if specified), at the first line otherwise. If item-id is omitted, the current input buffer parameter is used. Buffers are unchanged; files stay open. Transfer is one-way - control is not returned to original Proc. Any number of Procs of the same type may be chained.

3.2 []

[*filespec* {*item-id*}] {*label*}

[ ] calls an external Proc subroutine. Execution starts at label within the external subroutine (if specified), at the first line otherwise. If the item-id is omitted, the current input buffer parameter is used. When **RTN** is encountered, control returns to original Proc. Execution terminates if no more statements remain in the called Proc. Buffers remain unchanged and open files stay open. Any number of subroutines of the same type (PQ or PQN) may be nested. See also **GOSUB**.

3.3 +

+*n*

Adds integer n to current parameter in active input buffer. If no. of characters in parameter increases after a + command, other parameters are moved to the right. If a value is preceded by a plus sign, that sign is replaced with a zero. If buffer pointer is at end of buffer, a new parameter is created. If referenced parameter is nonnumeric, zero is used.

3.4 -

-*n*

Subtracts integer n from current parameter in input buffer. If no. of characters in parameter decreases after a - command, leading zeros are used to keep the same number of characters as before. Values within buffer can be preceded by a minus sign. If buffer pointer is at end of buffer, a new parameter is created. If referenced parameter is nonnumeric, zero is used.

3.5 A

A{*c*}

or **A**{*c*}*p*

or **A**{*c*}(*n*)

or **A**{*c*}(,*m*)

or **A**{*c*}(*n*,*m*)

Copies parameter at buffer pointer, or specified parameter, from active input to active output buffer.

Syntax elements	Description
c	character to enclose copied string: any nonnumeric character except left parenthesis ((). If backslash (\) used, value is concatenated with previous parameter in POB. If destination is SOB, *c* is ignored.
p	parameter to copy.
n	starting column number in PIB to begin copy. Copies to end of parameter or *m* characters.
m	number of characters to copy, from current PIB pointer or column *n*. Copy stops if attribute mark or end of buffer found.

3.6 B

B

Moves active input buffer pointer back to previous attribute mark. If on first character of parameter, pointer moved to attribute mark preceding previous parameter. If no attribute mark found, pointer positioned at start of input buffer. If pointer is already at start of buffer there is no change.

BO

Moves active output buffer pointer back one parameter and erases last parameter. If POB is active, buffer pointer moves backward until it finds an attribute mark or start of buffer, erasing the last parameter. If SOB active, **BO** clears entire buffer.

3.7 C

C or *****

Includes comment line in Proc (ignored at execution).

Caution

Do not confuse C command with C option of T command (which is used to clear screen). Do not put comments following FB, F-OPEN, F-READ, or F-UREAD, on continuation lines of T or L or on any line with an O, P, H or X command (see Multiple Commands for other exclusions).

3.8 D

D{*n*}{**+**}

Displays all or part of PIB or, used alone, current parameter of active input buffer. (Pointer not moved.)

Syntax elements	Description
n	specifies parameter of PIB to display. If n is 0, displays all parameters in PIB.
+	holds cursor at end of display by inhibiting RETURN/LINEFEED.

3.9 F

F

Moves input buffer pointer forward to next attribute mark. If no attribute mark found, pointer is positioned immediately after end of buffer.

F;*elem*{;*elem*}...;**?**[**P** | *r*]

Performs arithmetic calculations (using a stack processor), processing from left to right.

Syntax elements	Description
elem	operators and operands to be processed: + Adds top two stack entries and puts result at top. Successive values are moved up. - Subtracts stack 1 from stack 2 (opposite of English) and puts result in stack 1. Successive values move up. + Adds top two stack entries and puts result at top. Successive values are moved up. - Subtracts stack 1 from stack 2 (opposite of English) and puts result in stack 1. Successive values move up. r Direct or indirect buffer reference or select register containing value to place on stack. {C}n Stack constant numeric value specified by n.
?P	moves result of the calculation (top of stack) into parameter pointed to by PIB buffer pointer.
?r	moves result of calculation to location specified by r, direct or indirect buffer reference or select register.

F-C{**LEAR**} *fb-num*

Clears specified file buffer.

Syntax elements	Description
fb-num	file buffer number; direct or indirect reference.

F-D{**ELETE**} *fb-num*

Deletes item specified in attribute 0 of file buffer from file opened by **F-OPEN**. Releases any item lock set. File buffer contents not altered.

Syntax elements	Description
fb-num	file buffer number; direct or indirect reference.

F-F{**REE**} {*fb-num* {*item-id* }}

Releases item lock set by **F-UREAD**.

Syntax elements	Description
fb-num	file buffer number; direct or indirect reference. If not specified, releases all item locks set by this Proc in current context level.
item-id	item to be unlocked specified by direct or indirect reference to buffer or select register containing item name. If not specified, item-id in attribute zero is unlocked.

F-K{**LOSE**} *fb-num*

Closes specified file buffer.

Syntax elements	Description
fb-num	file buffer number; direct or indirect reference.

F-O{**PEN**} *fb-num filespec*

alt-cmd

Clears file buffer and opens a file for reads and writes. File remains open after Proc to Proc transfers.

Syntax elements	Description
fb-num	file buffer number (1 to 18) to which file is assigned.
alt-cmd	command(s) to be executed if open fails (must be on line following **F-OPEN**).

F-R{**EAD**} *fb-num item-id*

alt-cmd

Reads item from file into file buffer. If file to be read has not been opened, Proc terminates.

Syntax elements	Description
fb-num	file buffer number; direct or indirect reference.
alt-cmd	command(s) to be executed if item cannot be read (must be on line following **F-READ**).

F-UR{**EAD**} *fb-num item-id*

alt-cmd

Reads item from file into file buffer and sets item lock. If item is already locked, execution is suspended until item lock released. Item lock can be released by **F-DELETE**, **F-WRITE** or **F-FREE** command or when Proc terminates, or by another process (eg DataBasic or EDITOR).

Syntax elements	Description
fb-num	file buffer number; direct or indirect reference.
alt-cmd	command(s) to be executed if item cannot be read (must be on line following **F-UREAD**).

F-W{**RITE**} *fb-num*

Writes contents of specified file buffer as item on disk. (Item-id assumed to be in attribute zero of file buffer.) Any item lock set is released. File buffer contents not changed.

Syntax elements	Description
fb-num	file buffer number; direct or indirect reference.

FB *filespec* {*item-id* }

alt-cmd

Reads item into 'fast' buffer without first opening file. Previous 'fast' buffer contents are destroyed. If item-id is omitted, value at current active input buffer pointer location is used. The fast buffer is referenced as **&0**.a, where a is an attribute number. This can also be used as a 'scratch' buffer (without using **FB**). Note that '.'and 0 in the reference can be omitted giving compatibility with earlier releases.

Syntax elements	Description
alt-cmd	command(s) to be executed if named file cannot be opened (must be on line following **FB**).

3.10 G

G{**O**{**TO**}} *label*

Unconditionally transfers program control to a command line within the Proc beginning with *label*.

Syntax elements	Description
label	numeric label.

G{**O**} **B**

Transfers control to statement following last **M** (mark) command executed.

G{**O**} **F**

Transfers control to statement following next **M** (mark) command found.

GOSUB *label* or **[]** *label*

Transfers control to labelled subroutine (**RSUB** in subroutine returns control to line after this command).

Syntax elements	Description
label	numeric label.

3.11 H

Htext-str

Inserts text string in active output buffer, allowing execution of TCL verbs via the **P** command.

Text is inserted at current position of buffer pointer. The buffer pointer is then moved to end of output buffer, after the last character in inserted text.

If destination is POB, RETURN is supplied automatically. A series of one or more blanks is replaced by a single attribute mark. (Blanks within quotes are retained; everything within quotes is taken as a single attribute.) Prior to processing of TCL command, attribute marks are replaced with blanks. To concatenate the string with an existing parameter, omit leading blank.

If destination is SOB, each line of text must be terminated by including < to represent RETURN. A RETURN is not supplied automatically. One or more blanks are replaced by a single attribute mark. Prior to processing of SOB, AMs are replaced with blanks.

Syntax elements	Description
text-str	text to be placed in active output buffer: either a literal (not enclosed in quotes), a SYSTEM function, a direct or indirect reference to buffer, or select register containing character string.

3.12 I

IBH*text-str*

or

IBH*text-str*;*input-conv*;

or

IBH*text-str*:*output-conv*:

Places text in active input buffer. Like **IH**, but retains blanks instead of substituting attribute marks.

Syntax elements	Description
text-str	text to be placed in input buffer: either a literal (not in quotes), a SYSTEM function, or a buffer reference (do not use value marks or subvalue marks).

IBN{*c*}

Prompts for input from terminal and places input data in SIB. Like **IN**, but keeps input as a single parameter, retaining embedded blanks.

Syntax elements	Description
c	prompt character to remain in effect until reset by **IBN**, **IBP**, **IN**, or **IP** command. (Default is last prompt used or colon (:) if Proc was invoked from TCL.).

IBP{*c*{*r*}}

Prompts for input from terminal. Like **IP**, but keeps input as single parameter, retaining embedded blanks.

Syntax elements	Description
c	prompt character to remain in effect until reset by **IBN**, **IBP**, **IN**, or **IP** command. (Default is last prompt used or colon (:) if Proc was invoked from TCL.).
r	direct or indirect reference to buffer or select register where input data is to be placed.

IF{**N**} {**#**} *ref cmd* or **IF**{**N**} *ref oper expr cmd*

IF *ref cmd* tests for existence of a value. **IF #** *ref cmd* tests for absence of a value. Otherwise parameter given by *ref* is compared with *expr*. If condition is true, *cmd* is executed. If condition is false, *cmd* is skipped and processing continues with next line. **N** performs numeric (instead of string) comparisons. (See following pages for other **IF** command formats).

Syntax elements	Description
ref	can be one of following: **A** param Valid A command, not including c (surround character). It does not move a value but specifies input buffer parameter to test. *r* Direct / indirect ref. to buffer or select reg, or a SYSTEM function.
oper	can be = (equal) or # (not =). If N used, < (less), > (greater), [(less or =) or] (greater or =) are also valid.
expr	expression with which comparison is made. One of: *r* Direct / indirect ref. to buffer or select reg. *string* One or more non-blank chars. (invalid with N). *pattern* 'Format' string in parentheses that string being tested must match, consisting of one or more of the following elements (not valid with **N** or operators other than **=** and **#**.): (*n***A**) - *n* alphabetic characters. (*n***N**) - *n* numeric characters. (*n***X**) - *n* alphanumeric characters. (*string*) - non-numeric character(s).

Syntax elements	Description
	If *n* = 0, any number of characters of right type will match. Any number of match specifications can be combined, within parentheses to define a complete pattern.

IF{**N**} *ref* = *expr*]*expr*{]*expr*}... *cmd*]*cmd*{] *cmd* }...

or

IF{**N**} *ref* = *expr*]*expr*{]*expr*}... **G**{**O**} *label*] *label*{]*label* }...

or

IF{**N**} *ref* **#** *expr*]*expr*{]*expr*}... *cmd*

Compares ref with two or more expressions or values separated by value marks (CTRL+]), then conditionally executes one command. (If condition is false, cmd is skipped and processing continues with next line.)

With = operator (logical **OR**), if ref is equal to any expr, the condition is true and corresponding cmd is executed (subsequent expr s are not tested). With **#** operator (logical **AND**), if ref is not equal to all expr s, condition is true and cmd is executed.

Syntax elements	Description
N	performs numeric (instead of string) comparisons.
ref	can be one of the following: **A** *param* Valid **A** command, not including c (surround character). It does not move a value alone but specifies input buffer parameter to test. *r* Direct / indirect ref. to buffer or select reg, or a SYSTEM function.
oper	can be = (equal) or # (not =). If N used, < (less), > (greater), [(less or =) or] (greater or =) are also valid.
expr	expression with which comparison is made. One of: *r* Direct / indirect ref. to buffer or select reg. *string* One or more non-blank chars. (invalid with N). *pattern* Format string enclosed in parentheses (invalid with N; see standard IF for format). **]** Represents value mark (CTRL+]).

IF {**#**} **E** *cmd* or **IF E** *oper err-id cmd*

Conditionally executes a Proc command based on presence or absence of a system message after processing a TCL command. IF E cmd tests for presence of a system message. IF # E cmd tests for absence of a system message. IF E oper err-id cmdtests whether system message identity is equal or not equal to err-id.

Syntax elements	Description
oper	can be = (equal) or # (not equal).
err-id	item-id of a message in ERRMSG file.

IF {**#**} **S** *cmd*

Conditionally executes a Proc command based on the presence or absence of an active select list.

IH*text*

or

IH*text-ref* ;*input-conv*;

or

IH*text-ref* :*output-conv*:

or

IH

or

**IH **

Places text string in active input buffer, nulls an existing parameter, or creates new null parameters. Buffer pointer position is unchanged. Leading and trailing blanks are removed from text. Each set of embedded blanks is replaced with an attribute mark, creating separate parameters. (Use IBH to retain blanks.) If backslash immediately follows IH, current parameter is nulled; if separated by one space, new null parameter is created at pointer position. Backslash in any other position is treated as text. If pointer is at start of an existing parameter, IH replaces it with the text. If pointer is in middle of parameter, IH inserts new parameter (without a leading attribute mark) within original parameter, starting at location of pointer. If pointer is at end of input buffer, IH creates new parameter there. Pointer now points to attribute mark preceding first new parameter.

Syntax elements	Description
text	one or more characters (not including value or subvalue marks). Can be a literal, a direct or indirect reference to buffer or select register containing text, or a SYSTEM function.
text-ref	direct or indirect reference to buffer or select register.
input-conv, output-conv	Input / output conversion to apply to text. See *English Reference Manual*.

IN{*c*}

Prompts for input from terminal and places it in SIB. Contents of SIB are destroyed and pointer is set to buffer start before inserting data. Leading and trailing blanks are removed and sets of embedded blanks are replaced with an attribute mark.

Syntax elements	Description
c	prompt character to remain in effect until reset by **IBN**, **IBP**, **IN**, or **IP** command. (Default is last prompt used or colon (:) if Proc was invoked from TCL.).

Note
Because SIB has fallen into disuse, use of **IN** is not recommended. See **IP** command.

IP{*c*{*r*}}

Prompts for and accepts input from terminal. Leading and trailing blanks are removed and sets of embedded blanks are replaced with an attribute mark, thereby replacing the current parameter in the input buffer with multiple parameters. (Use **IBP** to retain blanks.)

IP{*c* } places data in active input buffer. **IP***cr* places input in buffer or select register specified; prompt character is required. RETURN only in response to input prompt nulls parameter indicated by buffer pointers.

Syntax elements	Description
c	prompt character to remain in effect until reset by **IBN**, **IBP**, **IN**, or **IP** command. (Default is last prompt used or colon (:) if Proc was invoked from TCL.).
r	direct or indirect reference to buffer or select register where input data is to be placed.

IT{**C**}{**A**}

Transfers a tape record of up to 300 bytes into the PIB, destroying its current contents. Buffer pointer is left at buffer start. The data is read from tape exactly as it appears.

Syntax elements	Description
C	converts EBCDIC data to ASCII during transfer.
A	masks 8-bit ASCII characters to 7 bits (where the most significant bit is 0).

3.13 L

L *elem*{,*elem*}...

Produces formatted printer output. If ends with comma (,), then command can continue on subsequent lines; if last element is a plus sign (+), then RETURN/LINEFEED is not output, and output from next L command is appended to current line.

Syntax elements	Description
elem	text or format print commands may be one of the following: "*text* " Text to output (enclosed in double quotes). *n* Skips n lines before printing. *r* {;*input* ;} Prints value obtained by direct or indirect reference to a buffer or select register r. Optional English input conversion may be applied to the value prior to printing. *r* {:*output* :}

Syntax elements	Description
	Prints value obtained by a direct or indirect reference to a buffer or select register r. Optional English output conversion may be applied to the value prior to printing. (*c*) Sets current horizontal position to column number *c*. Can be a direct or indirect reference to a buffer or select register. **+** Inhibits RETURN/LINEFEED output at the end of an L command. Not valid with HDR. **C** Closes print file, forcing printing of accumulated output. Not valid with HDR. **E** Ejects to top-of-form. Not valid with HDR. **N** As only element in L command, forces printer output to terminal for debugging. **HDR** As first element in L command, generates page heading. **L** With **HDR**, produces LINEFEED in the heading. **P** With **HDR**, prints page number in heading. **T** With **HDR**, prints time and date in heading. **Z** With **HDR**, zeros page number in heading.

3.14 M

M

Marks location to which a **GO F** or **GO B** command transfers control. **M** must be executed for it to be valid. Further commands may appear on same line, separated by subvalue marks (CTRL+\), but **M** must be first command on the line.

MS

Moves entire contents of SIB to the parameter currently pointed to by PIB pointer. SIB is left null.

MV *destination source*{,*source*}...{,*{n}}{,_}

or

MV *destination source*{**source*}...

Copies data between input buffer, output buffers, file buffers and select registers. If more than one source, each copied into consecutive locations in destination.

Syntax elements	Description
destination	direct or indirect reference to buffer or select register (not select register if source is of form Xx or In).

Syntax elements	Description
source	direct or indirect reference to buffer or select register, optionally followed by an English input conversion enclosed in semicolons or an output conversion enclosed in colons, a string of zero or more characters enclosed in single or double quotes, or a single character expressed as **X***x* or **I***n*, where x is a hex number in the range 00 to FE, n is a decimal in the range 0 to 255.
,*	Copies all source parameters starting with specified parameter. Destination buffer or select register is truncated after last parameter is copied if this is the last source parameter.
,**n*	Copies n further source parameters, following (and in addition to) the specified parameter.
*	Asterisks which separate source values, concatenates source values into one attribute in the destination.
,_	Destination is truncated after source is copied.

If attribute or parameter number in *destination* is larger than its current number of attributes or parameters, null values are created to pad as required. If destination is input buffer, pointer is placed at start of moved string.

MVA *destination source*

Copies value from source to destination and stores it as a multivalue. New value is stored in ascending ASCII sequence. If source is multivalued, it is copied, as is, to the destination. This might create duplicate values and destroy the ascending sequence. If destination is an input buffer, buffer pointer is positioned at beginning of parameter specified by destination.

Syntax elements	Description
destination	direct or indirect reference to buffer or select register.
source	direct or indirect reference to buffer or select register, or literal (string of one or more characters).

MVD *destination source*

Deletes data from a multivalued attribute or parameter in destination buffer. If source exists more than once in destination, only first occurrence is deleted. If source does not exist, no change occurs. If source is multivalued, MVD has no effect. If destination is the active input buffer, pointer is positioned at start of destination parameter.

Syntax elements	Description
destination	direct or indirect reference to buffer or select register from which to delete data.

Syntax elements	Description
source	direct or indirect reference to buffer or select register that contains data to delete or literal (string of one or more characters).

Note
Values in destination must be in ascending ASCII sequence.

3.15 O

O{*text*}{**+**}

Outputs a text string to terminal.

Syntax elements	Description
text	one or more characters to be displayed.
+	inhibits RETURN/LINEFEED at end of string leaving cursor positioned to right of last character displayed.

3.16 P

P{**P**}{**H**}{**L***n*}{**X**}

Executes the command that has been built in POB. Control is passed to TCL and, upon completion of command, control returns to Proc for any further processing. Only TCL verbs should be processed.

Syntax elements	Description
P	displays the command then prompts to continue.
H	suppresses (**HUSH**es) any terminal output that would be generated by processed command.
L*n*	sets execution lock (from 0 to 255).
X	terminates Proc after command executes (not used with any other option).

3.17 R

RI or **RI***p* or **RI**(*n*)

Without a parameter, clears entire PIB and positions pointer at buffer start. With a parameter, clears entire PIB as follows, leaving pointer at end of buffer.

Syntax elements	Description
p	clears PIB from parameter p to end of buffer. Can be direct or indirect reference to buffer or select register.

Syntax elements	Description
(*n*)	clears PIB from column n to end of buffer.

RO

Clears both output buffers and selects POB as active. Buffer pointers are placed at start of each buffer.

RSUB {*n*}

Terminates execution of a local subroutine and returns control to statement following **GOSUB** command that called it.

Syntax elements	Description
n	control returns to *n*'th statement after **GOSUB**. (Default is 1.

RTN {*n*}

Terminates execution of external Proc subroutine and transfers control to the Proc that called it.

Syntax elements	Description
n	control returns to *n*'th statement after **[ ]** (default is 1).

3.18 S

S*p* or **S**(*n*)

Positions buffer pointer in active input buffer to specified parameter or to specified column.

Syntax elements	Description
p	number of parameter to set buffer pointer to. Can be direct or indirect reference to buffer or select register.
(*n*)	number of column to set buffer pointer to.

STOFF or **ST OFF**

Selects POB as active and turns stack off so that all data moved to an output buffer with a subsequent **H** command is placed in POB. Stack can be turned on or off at any point. On initial entry or after **P** or **RO**, both output buffers are empty, and stack is off.

STON or **ST ON**

Selects SOB as active output buffer and turns stack on so that all data moved to an output buffer via **MV**, **H**, or **A** command is placed in SOB.

Stack is used to hold all commands necessary to feed interactive processors such as EDITOR and DataBasic when they have been invoked by a command built in the POB. If successive commands are placed in the stack each command must be terminated by < to generate a RETURN.

@SYS{**TEM**} (*sys-element*)

or

@SYS{**TEM**} (*reference*)

or

@SYS{**TEM**} (*indirect_reference*)

Returns current state of database parameters. The SYSTEM function can be used in conjunction with the MV, T, IH, IBH, H and IF commands only.

Syntax elements	Description
reference or *indirect.reference*	buffer reference supplying argument sys-element.
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 (Reality 7.x returns typeahead count). 15 Returns options used with last TCL command 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.

Syntax elements	Description
	20 Returns 1 if program running is cataloged. For Proc, therefore, always returns 0. 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 following attributes: 1 System serial number. 2 Set to 0 (Reality 7.x returns firmware type). 3 Set to 0 (Reality 7.x returns firmware version). 4 1 if Wordmate allowed, 0 if not. 5 2047 (number of Reality 7.x ABS frames). 6 Maximum number of active processes. 7 -1 (on Reality 7.x, returns Session Manager's process number). 8 0 (on Reality 7.x or earlier, returns maximum FID). 9 Number of workspace frames. 10 Maximum process number. 11 1 if UK system, 0 if not. 12 Memory size in Kbytes. 13 System type: 3 for Reality on UNIX or Windows, 0,1 or 2 for proprietary Reality 7.x or earlier. 23 Returns status of BREAK key: 0 Enabled 1 Disabled by DataBasic (automatically re-enabled when program ends). 2 Disabled from TCL (can not be re-enabled from DataBasic). 3 Disabled from DataBasic and TCL. 24 Returns 1 if character echoing enabled. 25 Returns 1 if current process is a TIPH. 26 Returns current prompt character. 27 Returns 1 if running from a Proc. 28 Returns system privilege level (0, 1 or 2). 29 Returns system frame size (1024). 30 Returns 1 if pagination is in effect. 35 Returns number of language in use. 36 Returns 0 (Reality 7.x returns default collation table). 37 Returns thousands separator. 38 Returns decimal separator. 39 Returns money sign. 40 Returns name of executing program.

Syntax elements	Description
	41 Returns Reality release number. 43 Returns port number, where item locked by **READU**. 44 Returns system type: 3 for Reality on UNIX or Windows; 0,1 or 2 for proprietary Reality 7.x or earlier. 45 For Proc, returns 0. 46 Returns 1 if last item read was a D-pointer. 47 Returns 1 if currently in a transaction. 48 Returns CCI (Consistent Circuit Identifier). Returns -1 for TIPH or if CCI undefined. 49 Returns PLId (Physical Location Identifier). 50 Returns user-id. 51 Returns software user-id (can be multivalued). 52 Returns database name. 53 Returns zero. 54 For Proc, returns null. 55 Returns system time in milliseconds, accurate to nearest second. 56 Returns zero (Reality 7.x 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. 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.

3.19 T

T *elem* {,*elem*}...

Produces formatted terminal output and displays buffer values.

Syntax elements	Description
elem	one of the following: "*text* " Text to output (within single or double quotes). *r* {;*input* ;} Outputs value obtained by direct or indirect reference to buffer or select register r. English input conversion optional. *r* {:*output* :} Outputs value obtained by a direct or indirect reference to buffer or select register r. English output conversion optional. (*c*,*r*) Sets terminal cursor to column c and row r. Can be direct or indirect buffer references. (*c*) Sets cursor to column c of current row. (,*r*) Sets terminal cursor to row r of current line. **cn* Outputs character c n times. n may be direct or indirect reference to buffer/select register. (-*n*) Provides terminal independent cursor control or video effects. **+** Inhibits RETURN/LINEFEED at command end. **B** Sounds terminal bell. **C** Clears screen (outputs top-of-form). **D** Causes a delay. **I***r* Converts integer r, where 0<=r<=255, into ASCII. May be direct or indirect reference to buffer or select register. **L** Terminates loop started with T. Elements between T and L are executed 3 times. **S***n* Outputs n spaces. May be direct or indirect reference to buffer or select register. **T** Marks beginning of loop terminated by L. **U** Moves cursor up one line. **X***r* Converts hex value r, where 00<= r =<FF, to ASCII. May be direct or indirect reference to buffer or select register. "**[K**" Clears the rest of line. ([is ESCAPE character.) "**[J**" Clears the rest of the screen. ([is ESCAPE)

TR {**ON**} or **TR OFF**

Traces Proc execution: displays each command as it is executed. **TR** turns trace on, **TR OFF** turns it off. Resets execution locks.

3.20 X

X{*text* }{**+**}

Halts execution of a Proc and returns control to TCL.

Syntax elements	Description
text	text to be displayed.
+	RETURN will not be output and cursor remains at present position when control is returned to TCL.

Section 4: Appendix A PQN Commands Grouped by Function

Arithmetic Calculations	
+	F;
-	
Conditional Operations	
IF	IF S
IF E	IFN
Data Movement	
A	MVA
MS	MVD
MV	
Debugging	
*	TR
C	PP
D	
Disk File Input/Output	
F-C{LEAR}	F-R{EAD}
F-D{ELETE}	F-UR{EAD}
F-F{REE}	F-W{RITE}
F-K{LOSE}	FB
F-O{PEN}	
Exiting Proc	
()	
Input/Output	
IBN	IT
IBP	L
IN	O
IP	T

Input Buffer	
B	IH
F	RI
IBH	S
Jump and Branch	
[]	GOSUB
()	M
G{O{TO}}	RSUB
G{O} B	RTN
G{O} F	
Output Buffer Operations	
BO	STOFF
H	STON
RO	
Processing	
P	

Section 5: Appendix B TCL Commands Related to Proc

Proc-related TCL Commands

For more details of the following, including syntax, see the *TCL Quick Reference Guide*.

PQ-COMPILE

Compiles large ProcS for greater efficiency.

PQ-RESELECT

Executed from Proc POB to reset select register pointer to beginning of item-id list.

Note
Cannot be used directly from TCL.

PQ-SELECT

Executed from Proc to load a list, produced by **SELECT** or other TCL command, into a select register. **PQ-SELECT** must be placed in the stack.

Note
Cannot be used directly from TCL.

TRACE

Implements **TR** function of Proc from TCL.

TRANSABORT

Undoes all updates performed by current transaction and releases all item-locks set during the transaction.

TRANSEND

'Commits' all updates made and releases all item locks set during the transaction.

TRANSQUERY

Reports transaction status of current port.

TRANSTART

Marks start of transaction.

Section 6: Appendix C PQ Procs

6.1 Differences Between PQ and PQN Proc

- PQ uses a blank as a delimiter instead of the attribute mark used in PQN.
- In commands that reposition the buffer pointer, PQ leaves the pointer set to the first character of the parameter. In PQN the pointer is set to an attribute mark.
- Buffer referencing is not available in PQ.

6.2 PQN Commands Not Available in PQ

F;	F-UR{EAD}	MS
F-C{LEAR}	F-W{RITE}	MV
F-D{ELETE}	FB	MVA
F-F{REE}	IBH	MVD
F-K{LOSE}	IBN	
F-O{PEN}	IBP	
F-R{EAD}	L	

6.3 Commands with the Same Functionality in PQ and PQN

()	IF	RSUB
[ ]	IF(mask)	RTN
+	IF E	S
-	IFN	STOFF
C	M	STON
D	O	TR
G	P	U
G{O} B	RI	X
G{O} F	RO	
GOSUB		

6.4 Commands with Differing Functionality in PQ and PQN Proc

Command	PQN Function	PQ Function
A	Moves from one attribute mark to the next, including embedded blanks.	Moves from first nonblank to next blank.
B	Moves input buffer pointer to previous attribute mark.	Moves input buffer pointer to first character of previous parameter.
BO	Moves output buffer pointer to previous attribute mark.	Moves output buffer pointer to first character of previous parameter.
F	Moves input buffer pointer forward to next attribute mark.	Moves input buffer pointer forward to first character of next parameter.
H	Moves character string into current output buffer location, replacing each set of blanks with an attribute mark.	Moves character string into current output buffer location, with blanks preserved.
IH	Moves specified string into current input buffer location, replacing each set of blanks with an attribute mark.	Moves specified string into current input buffer location, with blanks preserved (like PQN IBH).
IP	Moves specified string into current input buffer location, with blanks preserved (like PQN IBH).	Moves input data into current input buffer location with blanks preserved (like PQN IBP).

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