# Table of Contents

## About This Book

Symbols ................................................................................................................................. 3

PQ Declarator ......................................................................................................................... 7

PROC Commands .................................................................................................................... 9

Symbols Used in Caché MultiValue PROCs ................................................................. 4

PQ, PQN, PQX ....................................................................................................................... 8

+ ........................................................................................................................................ 10

* ........................................................................................................................................ 11

( ......................................................................................................................................... 12

[ ....................................................................................................................................... 13

A ......................................................................................................................................... 14

B ......................................................................................................................................... 16

BO ...................................................................................................................................... 17

C ......................................................................................................................................... 18

D ......................................................................................................................................... 19

F ......................................................................................................................................... 20

FB, FBU ............................................................................................................................. 21

F-CLEAR ............................................................................................................................ 22

F-DELETE .......................................................................................................................... 23

F-FREE ............................................................................................................................... 24

F-KLOSE ............................................................................................................................. 25

F-OPEN .............................................................................................................................. 26

F-READ .............................................................................................................................. 27

F-UREAD ........................................................................................................................... 28

F-WRITE ............................................................................................................................. 29

GO ..................................................................................................................................... 30

GOSUB ................................................................................................................................. 31

H ......................................................................................................................................... 32

IF, IFN ................................................................................................................................. 33

IH, IBH ............................................................................................................................... 35

IN, IBN ............................................................................................................................... 36

IP, IBP ................................................................................................................................. 37

IS, IBS ................................................................................................................................. 38

M ......................................................................................................................................... 39

MV ...................................................................................................................................... 40

MVA ................................................................................................................................. 41

MVD ................................................................................................................................... 42

O ......................................................................................................................................... 43

P, PH, PP, PW, PX .......................................................................................................... 44

Q ......................................................................................................................................... 46

RI ....................................................................................................................................... 47

RO ..................................................................................................................................... 48

RSUB ................................................................................................................................. 49

S ......................................................................................................................................... 50

SP ...................................................................................................................................... 51

SS ...................................................................................................................................... 52
About This Book

This book provides reference material for the PROC (procedure) commands of the Caché MultiValue implementation.

This book contains the following sections:

- 
- 

There is also a detailed Table of Contents.

Other related topics in the Caché documentation set are:

- Using the MultiValue Features of Caché
- Operational Differences between MultiValue and Caché
- The Caché MultiValue Spooler

For general information, see Using InterSystems Documentation.
Symbols
# Symbols Used in Caché MultiValue PROCs

A table of characters used in Caché MultiValue PROCs as operators, etc.

## Table of Symbols

The following are the literal symbols used in Caché MultiValue PROCs. (This list does not include symbols indicating format conventions, which are not part of the language.)

The name of each symbol is followed by its ASCII decimal code value.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name and Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>[space] or [tab]</td>
<td><em>White space (Tab (9) or Space (32)): One or more whitespace characters between keywords, identifiers, and variables.</em></td>
</tr>
<tr>
<td>!</td>
<td><em>Exclamation Mark (33): A reference specifying a select list. For example: !6.</em></td>
</tr>
<tr>
<td>&quot;</td>
<td><em>Double Quote (34): A delineator used to enclose a string literal.</em></td>
</tr>
<tr>
<td>#</td>
<td><em>Pound (35): A reference specifying the active output buffer.</em></td>
</tr>
<tr>
<td>%</td>
<td><em>Percent sign (37): A buffer reference specifying the primary input buffer (PIB). For example, %2.</em></td>
</tr>
<tr>
<td>&amp;</td>
<td><em>Ampersand (38): A reference specifying a numbered file buffer: &amp;2 = file buffer 2. &amp;2.4 = record 4 in file buffer 2.</em></td>
</tr>
<tr>
<td></td>
<td><em>A reference specifying the fast file buffer: &amp; = fast file buffer. &amp;2 = record 2 in the fast file buffer.</em></td>
</tr>
<tr>
<td>`</td>
<td><em>Single Quote (39): A delineator used to enclose a string literal.</em></td>
</tr>
<tr>
<td>( )</td>
<td><em>Parentheses (40,41):</em> In IF conditional, encloses a pattern match code.*</td>
</tr>
<tr>
<td>(</td>
<td><em>Left Parenthesis (40): The ( command.</em></td>
</tr>
<tr>
<td>*</td>
<td><em>Asterisk (42): In MV command, a wildcard specifying moving all remaining fields.</em></td>
</tr>
<tr>
<td>+</td>
<td><em>Plus sign (43): The + command.</em></td>
</tr>
<tr>
<td>,</td>
<td><em>Comma (44): For the T command, a separator character for a list of arguments. For the MV command, a separator character for a series of source values.</em></td>
</tr>
<tr>
<td>-</td>
<td><em>Minus sign (45): The – command.</em></td>
</tr>
<tr>
<td>.</td>
<td><em>Period (46):</em></td>
</tr>
<tr>
<td>:</td>
<td><em>Colon (58):</em></td>
</tr>
<tr>
<td>Symbol</td>
<td>Name and Usage</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
</tr>
<tr>
<td>;</td>
<td>Semicolon (59):</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than (60): In IF conditional, less than.</td>
</tr>
<tr>
<td>=</td>
<td>Equal sign (61): In IF conditional, equal to.</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than (62): In IF conditional, greater than.</td>
</tr>
<tr>
<td>[]</td>
<td>Square Brackets (91 &amp; 93): The [ command.</td>
</tr>
<tr>
<td>[</td>
<td>Left Square Bracket (91): The [ command.</td>
</tr>
<tr>
<td>\</td>
<td>Backslash (92): A delineator used to enclose a string literal.</td>
</tr>
<tr>
<td></td>
<td>A display delineator used to represent the @SM character.</td>
</tr>
<tr>
<td>]</td>
<td>Right Square Bracket (93): A display delineator used to represent the @VM character.</td>
</tr>
<tr>
<td>^</td>
<td>Caret (94): A display delineator used to represent the @AM character.</td>
</tr>
<tr>
<td>_</td>
<td>Underscore (95): A display delineator used to represent the @FM character.</td>
</tr>
</tbody>
</table>
The first line of a PROC.

PQ [comment]
PQN [comment]
PQX [comment]

Description
The first line of a Caché MultiValue PROC must contain the PQ, PQN, or PQX declarator. These keywords specify the type of PROC and its compatibility with different MultiValue emulations.

This declarator can be followed by a descriptive comment containing any characters.

- **PQ** — Caché PQ procs always use @AM as the input buffer delimiter. MVBasic PROCREAD and PROCWRITE convert these @AM delimiters to spaces.
- **PQN** — If an emulation doesn't support PQN (D3, for example), the system compiles the PQN proc as a PQ proc.
- **PQX** — Supported for compatibility with D3 and MVBase.

Emulation
PQ and PQN PROCS use @AM as the buffer delimiter for INFORMATION, IN2, PICK, PIOpen, and UniVerse emulations. Other emulations use a blank space as the buffer delimiter.

See Also
- **T** PROC command
PROC Commands
**+**

Adds an integer to a field in the input buffer.

<table>
<thead>
<tr>
<th>+n</th>
</tr>
</thead>
</table>

**Arguments**

<table>
<thead>
<tr>
<th>n</th>
<th>An integer value.</th>
</tr>
</thead>
</table>

**Description**

The + PROC command adds \( n \) to the numeric value field pointed to by the input buffer pointer. If the input buffer pointer is not pointing to a field, or is pointing to a non-numeric value field, no operation occurs. If \( n \) is not specified, or is a non-integer value, no operation occurs.

The - PROC command subtracts \( n \) from the numeric value field pointed to by the input buffer pointer.

**See Also**

- - PROC command
- D PROC command
Subtracts an integer from a field in the input buffer.

\[-n\]

**Arguments**

| n  | An integer value. |

**Description**

The `-` PROC command subtracts \( n \) from the numeric value field pointed to by the input buffer pointer. If the input buffer pointer is not pointing to a field, or is pointing to a non-numeric value field, no operation occurs. If \( n \) is not specified, or is a non-integer value, no operation occurs.

The `+` PROC command adds \( n \) to the numeric value field pointed to by the input buffer pointer.

**See Also**

- `+` PROC command
- `D` PROC command
( [DICT] filename [procname] [] ] [label]

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>The name of the file that contains the target PROC. <em>filename</em> can be a file name, or a reference to a buffer or a select list that contains the file name.</td>
</tr>
<tr>
<td>procname</td>
<td>The name of the target PROC within the specified file. <em>procname</em> can be a PROC name, or a reference to a buffer or a select list that contains the PROC name.</td>
</tr>
<tr>
<td>label</td>
<td><em>Optional</em> — An integer specifying a label within the called PROC to jump to. If not specified, execution begins at the first line of the PROC. The optional closing parenthesis is required when specifying a <em>label</em>. For further details on labels, refer to the GO command.</td>
</tr>
</tbody>
</table>

**Description**

The ( PROC command calls the specified PROC. It does not return to the invoking PROC upon completion. To call a PROC and then return to the invoking PROC, use the [ command.

The optional DICT keyword specifies that the PROC is stored in the file dictionary. The DICT keyword is not case-sensitive.

**See Also**

- [ PROC command ]
Transfers execution to another PROC, then returns.

[[DICT] filename [procname] []] [label]

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>The name of the file the contains the target PROC. <em>filename</em> can be a file name, or a reference to a buffer or a select list that contains the file name.</td>
</tr>
<tr>
<td>procname</td>
<td>The name of the target PROC within the specified file. <em>procname</em> can be a PROC name, or a reference to a buffer or a select list that contains the PROC name.</td>
</tr>
<tr>
<td>label</td>
<td><em>Optional</em> — An integer specifying a label within the called PROC to jump to. If not specified, execution begins at the first line of the PROC. The optional closing bracket is required when specifying a <em>label</em>. For further details on labels, refer to the GO command.</td>
</tr>
</tbody>
</table>

**Description**

The [ PROC command calls the specified PROC. Upon completion, execution returns to the invoking PROC. To call a PROC without returning to the invoking PROC, use the ( command.

The optional DICT keyword specifies that the PROC is stored in the file dictionary. The DICT keyword is not case-sensitive.

**See Also**

- ( PROC command
A

Copies a field from the input buffer to the output buffer.

A
A
A, nnn
Achar [nnn]
Achar (nnn, nnn)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nnn</td>
<td>Optional — An integer specifying the number of characters in the input buffer to copy, starting from the current buffer pointer position. If nnn is omitted, or if nnn equals the number of characters in the field, the entire field is copied, and the current buffer pointer is advanced to the next parameter.</td>
</tr>
<tr>
<td>char</td>
<td>Optional — A single character applied as the delimiter character before and after an output data value. Common char values are the ' (single quote) and &quot; (double quote) characters. If no char is specified, the default is to delimit an output data value with blank spaces. Use \ as the char value to specify that no delimiter should be applied. char is applied when copying to the primary output buffer (POB); char is ignored when copying to the secondary output buffer (SOB).</td>
</tr>
</tbody>
</table>

Description

The A PROC command copies the specified nnn number of characters from the input buffer to the active output buffer. In Caché and most emulations, Annn is identical to A,nnn (see below for exceptions). A without nnn copies the entire current field from the active input buffer to the active output buffer.

When the primary output buffer (POB) is active, A copies the specified data from the input buffer to the POB, appending the copied data to the existing data at the current buffer pointer position. It delimits the output value by a separator character before and after the data value. The default separator character is a blank space. This default delimiting with blanks occurs in PQ procs; it does not occur in PQN procs. To output the data value without delimiter characters, use A\.

A with no arguments copies the current field from the active input buffer to the primary output buffer. When copying to the POB, it delimits the output field value with blank spaces. If there is no field value in the input buffer, A outputs a blank space by default.

A recognizes the semicolon (;) in the input buffer as a separator between values.

When the secondary output buffer (SOB) is active, A copies to the secondary output buffer, appending the copied data to the existing data at the current buffer pointer position. It copies the specified data from the input buffer exactly; no delimiting separator characters are added. Use the STON command to activate the secondary output buffer.

The A command maintains an input buffer pointer so that repeated invocations output successive substrings of a field. If nnn is omitted or equals the number of characters in the field, the entire field is returned, and the A pointer is advanced to the next field in the input buffer.

Commonly, the space between the A command name and the first argument is omitted.

The A command is frequently used as a reference for IF and GOTO. The IF command condition clause can take an A reference. The G (GOTO) command can take an A reference to specify retrieval of a label location from the input buffer. When used with IF or G, A cannot contain a char delimiter character.
**Emulation**

In UniVerse, if the input string already has delimiters, the A command does not add additional delimiters.

In REALITY, the A,nnn syntax uses the comma (,) as the char delimiter character. In all other emulations, the comma is ignored.

In jBASE, if nnn equals the length of the data, A does not advance the input buffer pointer to the next field. In Caché and all other emulations, A does advance the input buffer pointer to the next field.

In jBASE and REALITY, for a PQN proc, the A command does not recognize a semicolon (;) in the input buffer as an element separator.

**See Also**

- STON PROC command
- T PROC command
Moves the input buffer pointer backwards.

**Arguments**

None.

**Description**

The **B** PROC command moves the input buffer pointer backwards to the previous field, as indicated by a @FM field mark character.

The **F** PROC command moves the input buffer pointer forward to the next field. The **BO** PROC command moves the output buffer pointer backwards to the previous field.

**See Also**

- **BO** PROC command
- **F** PROC command
BO

Moves the output buffer pointer backwards.

Arguments

None.

Description

The BO PROC command moves the output buffer pointer backwards to the previous field.
The B PROC command moves the input buffer pointer backwards to the previous field.

See Also

- B PROC command
C

Specifies a single-line comment.

C [text]

Arguments

| text | Optional — A comment. |

Description

The C PROC command allows you to include a single-line comment in a PROC.
**D**

Displays a field from the input buffer.

\[ D \{f[,nnn]\} \]

**Arguments**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( f )</td>
<td>A field number in the input buffer, specifying which field to display. If ( f=0 ), display the entire input buffer. ( f ) can be specified as an integer, or as a reference to a buffer or a select list that contains the integer.</td>
</tr>
<tr>
<td>( nnn ) (Optional)</td>
<td>An integer specifying the number of characters in ( f ) to display, starting from the current position. If ( nnn ) equals the number of characters in the field, the entire field is displayed, and the ( D ) pointer is positioned to the next field. If ( nnn ) is omitted, all of the characters in the field are displayed.</td>
</tr>
</tbody>
</table>

**Description**

The **D** PROC command is used to display a parameter or a specified substring of a parameter. For the purpose of display, **D** strips leading blanks from the field value(s).

Commonly, the space between the command name and the text is omitted, as shown in the following example:

\[ D3 \]

**Emulation**

In UniData emulation, **D** strips quote characters from the beginning and end of the displayed parameter.

**See Also**

- T PROC command
F

Moves the input buffer pointer forward.

**Arguments**

None.

**Description**

The F PROC command moves the input buffer pointer forward to the next field, as indicated by a @FM field mark character. The B PROC command moves the input buffer pointer backwards to the previous field.

**Emulation**

In jBASE emulation, F extends the buffer when moved beyond the current contents of the buffer.

**See Also**

- B PROC command
FB, FBU

Opens a file and reads a record into the fast file buffer.

```
FB [([DICT] filename [itemId][])
FBU [([DICT] filename [itemId][])
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>The name of the file to read into the fast file buffer. <code>filename</code> can be a literal, or a reference to a buffer or a select list that contains the filename.</td>
</tr>
<tr>
<td>itemId</td>
<td>Optional — The item ID of the item to be read into the fast file buffer. <code>itemId</code> can be a literal, or a reference to a buffer or a select list that contains the item ID. If you do not specify an item ID, FB selects the current item.</td>
</tr>
</tbody>
</table>

**Description**

The FB PROC command opens a file and reads a single record. This record is read into the fast file buffer. The fast file buffer is referenced as & or &0. This command has two forms, FB (simple read) and FBU (read with update lock). A subsequent FB command overwrites the contents of the fast file buffer.

You can use the optional DICT keyword to specify that the file is a dictionary file. The DICT keyword can be supplied as a literal, or indirectly. The enclosing parentheses are optional in Caché MultiValue. They are required in some MultiValue implementations and are not permitted in other MultiValue implementations.

FB is functionally equivalent to an F-OPEN followed by an F-READ. FBU is functionally equivalent to an F-OPEN followed by an F-UREAD. An item locked with FBU can be unlocked using F-FREE.

This is a PQN command.

**See Also**

- F-FREE PROC command
- F-OPEN PROC command
- F-READ PROC command
- F-UREAD PROC command
**F-CLEAR**

Clears the file buffer.

| F-CLEAR fb | F-C fb |

**Arguments**

| fb | An integer specifying the file buffer. Available values are 1 through 9. |

**Description**

The **F-CLEAR** PROC command is used to clear the file buffer and the item Id. The fb file buffer is assigned using the F-OPEN command.

F-CLEAR is a PQN command.

F-CLEAR clears file buffers (&2). RI clears input buffers (%2). RO clears output buffers (#2).

**See Also**

- F-OPEN PROC command
- F-UREAD PROC command
- F.WRITE PROC command
- RI PROC command
- RO PROC command
F-DELETE

Deletes the file buffer oref.

F-DELETE fb
F-D fb

**Arguments**

| fb  | An integer specifying the file buffer. Available values are 1 through 9. |

**Description**

The **F-DELETE** PROC command deletes the current record. It does this by deleting the object reference (oref) for the record contained in the file buffer. The file must have been opened using the **F-OPEN** command, which assigned it a numbered *fb* file buffer. **F-DELETE** does not change the contents of the file buffer.

**F-DELETE** is a PQN command.

**Example**

The following example show the **F-DELETE** command. It writes a record to the file buffer, then deletes the record value:

```pqn
PQN
F-OPEN 7 VOC
MV %? "MyData"
F-WRITE 7
F-DELETE 7
```

**See Also**

- **F-OPEN** PROC command
- **F-WRITE** PROC command
**F-FREE**

Frees the lock on a specified item in the file buffer.

```plaintext
F-FREE [fb [itemId]]
F-F [fb [itemId]]
```

**Arguments**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fb</code></td>
<td>An integer specifying the number assigned to the file buffer by the F-OPEN command. Available values are 1 through 9. If omitted, uses the currently open file buffer.</td>
</tr>
<tr>
<td><code>itemId</code></td>
<td>The item ID of the item to be unlocked. <code>itemId</code> can be specified as an integer, or as a reference to a buffer or a select list that contains the <code>itemId</code> value.</td>
</tr>
</tbody>
</table>

**Description**

The **F-FREE** PROC command frees the update lock on an item in the file buffer. An update lock is applied by the **F-UREAD** command. If you specify no `itemId`, **F-FREE** frees any locked items in the file buffer.

**F-FREE** is a PQN command.

**See Also**

- **F-UREAD** PROC command
F-KLOSE

Closes the file buffer.

F-KLOSE fb
F-K fb

Arguments

| fb | An integer specifying the file buffer. A value of 0 closes the fast file buffer. |

Description

The F-KLOSE PROC command closes the file buffer. A non-zero numbered fb file buffer is assigned using the F-OPEN command. A fb value of 0 closes the fast file buffer. (The fast file buffer is activated by the FB command.)

This command is not commonly used (file buffers are closed when the PROC terminates). It is not supported in all emulations; it is provided here primarily for compatibility with REALITY applications.

F-KLOSE is a PQN command.

See Also

- F-OPEN PROC command
- FB PROC command
F-OPEN

Opens a file and assigns it to a file buffer.

F-OPEN fb [DICT] filename
F-O fb [DICT] filename

Arguments

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>fb</td>
<td>An integer specifying the file buffer to be used. Available values are 1 through 9.</td>
</tr>
<tr>
<td>filename</td>
<td>The name of the file to open. filename can be a literal, or a reference to a buffer or a select list that contains the file name.</td>
</tr>
</tbody>
</table>

Description

The F-OPEN PROC command opens a file and assigns it to a file buffer. You can use the optional DICT keyword to specify that the file is a dictionary file.

F-OPEN is a PQN command.

See Also

- F-FREE PROC command
- F-KLOSE PROC command
- F-READ PROC command
- F-UREAD PROC command
- F-WRITE PROC command
**F-READ**

Reads a record into a file buffer.

```
F-READ fb itemId
F-R fb itemId
```

**Arguments**

<table>
<thead>
<tr>
<th>fb</th>
<th>An integer specifying the file buffer associated with the file. Available values are 1 through 9.</th>
</tr>
</thead>
<tbody>
<tr>
<td>itemId</td>
<td>The item ID of the item to be read into the file buffer. <code>itemId</code> can be an integer, or a reference to a buffer or a select list that contains the item ID.</td>
</tr>
</tbody>
</table>

**Description**

The **F-READ** PROC command reads a specified item from a file into the file buffer. The file must have been opened using the **F-OPEN** command, which assigned it a numbered `fb` file buffer.

**F-READ** is a PQN command.

**Examples**

The following examples show the **F-READ** command. The first reads a literal value, the second reads a reference to a buffer containing the value.

```
PQN
F-OPEN 7 VOC
F-READ 7 MyData

PQN
F-OPEN 7 VOC
MV %2 "MyData"
F-READ 7 %2
```

**See Also**

- **F-OPEN** PROC command
- **F-UREAD** PROC command
- **F-WRITE** PROC command
**F-UREAD**

Reads a record into a file buffer and applies an update lock.

```
F-UREAD fb itemId
F-U fb itemId
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fb</code></td>
<td>An integer specifying the file buffer.</td>
</tr>
<tr>
<td><code>itemId</code></td>
<td>The item ID of the record to be read from the file buffer. <code>itemId</code> can be an integer, or a reference to a buffer or a select list that contains the item ID.</td>
</tr>
</tbody>
</table>

**Description**

The **F-UREAD** PROC command reads a specified record from a file into the file buffer. The file must have been opened using the **F-OPEN** command, which assigned it a numbered `fb` file buffer.

If the requested record is locked by another process, **F-UREAD** waits until the lock becomes available.

**F-UREAD** applies an update lock on the record, preventing other users access to the record. This lock is released using the **F-FREE** command.

**F-UREAD** is a PQN command.

**See Also**

- **F-FREE** PROC command
- **F-OPEN** PROC command
- **F-READ** PROC command
- **F-WRITE** PROC command
F-WRITE

Writes a record from the file buffer to the file.

### Arguments

| fb | An integer specifying the file buffer. Available values are 1 through 9. |

### Description

The **F-WRITE** PROC command is used to write the current item in the file buffer to the file.

**F-WRITE** is a PQN command.

### See Also

- T PROC command
**GO**

Go to a label.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nnn</td>
<td>An integer specifying a label. nnn can be specified as a label integer, or as a reference to a buffer or a select list that contains the label integer.</td>
</tr>
</tbody>
</table>

**Description**

The **GO** PROC command redirects execution to the specified label location. You can establish a label by specifying an integer at the beginning of a PROC line. This integer is separated by a blank space from a PROC command on the same line. You can specify up to 256 labels in a PROC. If a duplicate label occurs, **GO** goes to the first label with that number, searching from the beginning of the PROC.

**GO**, **G**, and **GOTO** are synonyms.

**GO** nnn goes to the specified label. **GOSUB** nnn goes to the specified label, then can return with **RSUB**. The ( command and [ command transfer execution to a different PROC, and can specify a label number within that PROC. **GO** F goes forward to the next location established by the **M** command. **GO** B goes backward to the most-recently encountered location established by the **M** command.

**GO** can take an A reference to specify retrieval of a label location from the input buffer. When used with **IF** or **GO**, A cannot contain a char delimiter character.

**Example**

The following example shows **GO** jumping to label 10:

```
PQN
MV %1 "A"
IF %1 = "A" GO 10
  Don't Display this
XEnd of proc not taken
10 ODisplay this
XEnd of proc taken
```

**See Also**

- **GOSUB** PROC command
- **M** PROC command
**GOSUB**

Go to a label with option to return.

```
GOSUB nnn
```

### Arguments

| nnn | An integer specifying a label. For further details on labels, refer to the GO command. |

### Description

The **GOSUB** PROC command redirects execution to the specified label location. You can then use **RSUB** to return to the line following the **GOSUB** command.

### Example

The following example shows **GOSUB** jumping to label 10, then **RSUB** returning to the **X** (exit) command following the **GOSUB**:

```
PQN
MV %1 "A"
IF %1 = "A" GOSUB 10
XEnd of PROC
ODon't Display this
10 ODisplay this
RSUB
```

### See Also

- **GO PROC command**
- **IF PROC command**
- **RSUB PROC command**
**H**

Adds a string to the output buffer.

**H text [< | <<]**

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>A string to append to the contents of the active output buffer. Text can be specified as a string literal, or as a reference to a buffer or a select list that contains the string literal. A reference must begin with a %, !, &amp;, or # character, followed by a number (%2), or followed by another of the four reference characters (%%). Otherwise, it is treated as a string literal. No string delimiters are required for a string literal.</td>
</tr>
</tbody>
</table>

**Description**

The H PROC command writes a field from a string to the active output buffer. H divides a string into fields, using the blank space as the field delimiter. However, blank spaces within a quoted substring are not treated as a field delimiter. The H command the resets the output buffer pointer to the end of the buffer contents.

The < character outputs a carriage return character if the secondary output buffer is active.

Commonly, the space between the command name and the text is omitted, as shown in the following example which writes an MVBasic SSELECT statement into the output buffer:

H SSELECT Car.File WITH STATUS "I"

**Emulation**

In Caché and UniVerse, << is ignored. In other MultiValue emulations, it outputs a < and a newline character.

**See Also**

- IH PROC command
- O PROC command
**IF, IFN**

Conditionally executes a command.

```
IF condition command
IF condition label
IFN condition command
IFN condition label
```

where condition:
- `x {= | # | < | >} y`
- `x {= | #} (patcode)`
- `[#]E`
- `[#]Sn`

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition</td>
<td>A condition that resolves to a boolean value. If TRUE (1), command is executed. If FALSE (0), command is not executed.</td>
</tr>
<tr>
<td>command</td>
<td>The command to execute if condition=1 (TRUE).</td>
</tr>
<tr>
<td>label</td>
<td>An integer specifying the location to go to if condition=1 (TRUE).</td>
</tr>
</tbody>
</table>

**Description**

The **IF** PROC command applies a boolean test on a condition statement, and, if the condition is true, either executes the specified command, or goes to the specified label location. A condition is true if any one of the value comparisons are true.

**IF** performs a string condition comparison. **IFN** performs a numeric condition comparison.

A condition can use the `=` (equal to) or `#` (not equal to) operator. The following conditions are supported:

- `x {= | #} y` An equality condition compares a value to a value. A value can be a literal, an A command, or a reference to a buffer or select list. When used with **IF** an A reference cannot contain a char delimiter character. If x is a reference, only the first value found in the buffer or select list is compared to y. If y is a reference, all of the values found in the buffer or select list are compared to x.
- `x {= | #} (patcode)` A pattern match condition compares a value to a pattern match code. A pattern match code is enclosed in parentheses; for example, `(3A)`. A=alphabetic characters; N=numbers. If x is a reference, only the first value found in the buffer or select list is matched to the pattern code.
- `[#]E` An error condition tests whether an error code exists (E) or does not exist (#E). Because E contains the error code value, you can also perform an **IF** test on the value of the error code: `E < 1`. Error codes are integers beginning with 260, through 277.
- `[#]Sn` A select list condition tests whether the specified select list is active. For example, `S3` determines if select list 3 is active; `#S3` determines if select list 3 is not active.

**Compound IF Expressions**

You can use the ``` character to create compound IF expressions. In a compound expression, the right side of the condition consists of two (or more) match conditions, each of which has its corresponding command or label.

In the following example, the value of x is first compared to all of the values in %3; if any of these match, command1 is executed. If x does not match %3, x is then matched with all of the values in %4; if any of these match, command2 is executed.

```
IF x = %3]S4 [command1]command2
```
In the following example, the value of \( x \) is first compared to the literal Foo; if this is a match, a goto operation is performed to label 100. If \( x \) does not match Foo, \( x \) is then matched with Bar; if this is a match, a goto operation is performed to label 200.

\[
\text{IF } x = \text{Foo|Bar 100|200}
\]

**See Also**

- A PROC command
- GO PROC command
- P PROC command
IH, IBH

Inserts a string into the active input buffer.

IH [text]
IBH text

Arguments

| text | Optional — A string to insert into the input file buffer. text can be specified as a string literal, or as a reference to a buffer or a select list that contains the string literal. A reference must begin with a %, !, &, or # character, followed by a number (%2), or followed by another of the four reference characters (%%). No string delimiters are required for a string literal. |

Description

The IH PROC command is used to insert a string into the active input buffer. An IH with no argument defaults to the empty string.

IH removes blanks within the input string. IBH retains blanks within the input string.

IH inserts text at the input buffer pointer location. IH does not move the input buffer pointer.

If the primary input buffer (PIB) is active, IH clears the secondary input buffer (SIB).

Commonly, the space between the command name and the text is omitted, as shown in the following example:

PQ
RI
IH10
F
IH20

This example resets the input buffer, then IH10 inserts the number 10 at the beginning of the input buffer. The F command advances the input buffer pointer, then IH20 inserts the number 20 as the second value in the input buffer.

In a PQN PROC, a backslash (\) in text clears the current input buffer element. In a PQ PROC, a backslash (\) in text is a literal character.

See Also

- F PROC command
- RI PROC command
IN, IBN

Reads input from the user terminal into the SIB.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN [prompt]</td>
<td>IN removes blanks within the input string.</td>
</tr>
<tr>
<td>IBN [prompt]</td>
<td>IBN retains blanks within the input string.</td>
</tr>
</tbody>
</table>

**Arguments**

- prompt
  
  *Optional* — The prompt character.

**Description**

The IN PROC command prompts the user for input and reads this input into an input buffer. It uses the secondary input buffer (SIB): setting it as active, clearing it, then receiving input into it. IN removes blanks within the input string. IBN retains blanks within the input string.

Commonly, user input placed in the secondary input buffer is use by the P command.

In Caché MultiValue, the IN and IS commands are identical.

**See Also**

- IP PROC command
- IS PROC command
- P PROC command
IP, IBP

Reads input from the user terminal into a specified buffer or select list.

**IP [prompt] reference**

**IBP [prompt] reference**

**Arguments**

<table>
<thead>
<tr>
<th>prompt</th>
<th>Optional — The prompt character.</th>
</tr>
</thead>
<tbody>
<tr>
<td>reference</td>
<td>A reference to a buffer or a select list. A reference must begin with a %, !, &amp; or # character, followed by a number (%2), or followed by another of the four reference characters (%%).</td>
</tr>
</tbody>
</table>

**Description**

The **IP** PROC command is used to read input from the user terminal into the specified buffer or select list.

**See Also**

- IN PROC command
- IS PROC command
- P PROC command
IS, IBS

Prompts for input.

<table>
<thead>
<tr>
<th>IS [prompt]</th>
<th>IBS [prompt]</th>
</tr>
</thead>
</table>

Arguments

<table>
<thead>
<tr>
<th>prompt</th>
<th>Optional — The prompt character.</th>
</tr>
</thead>
</table>

Description

The IS PROC command prompts the user for input and reads this input into an input buffer. It uses the secondary input buffer (SIB): setting it as active, clearing it, then receiving input into it. IS removes blanks within the input string. IBS retains blanks within the input string.

Commonly, user input placed in the secondary input buffer is use by the P command.

In Caché MultiValue, the IS and IN commands are identical.

See Also

- IN PROC command
- IP PROC command
- P PROC command
Marks a location in the PROC.

**Arguments**

None.

**Description**

The **M** PROC command is used to mark a location in the PROC. This location is used by the **GO F** and **GO B** commands.

**See Also**

- **GO PROC command**
MV

Moves a value into a buffer or select list.

MV target source
MV target,source
MV target=source

Arguments

| target | The buffer or select list that receives the contents of source. Can be a %n (input buffer), #n (output buffer), or !n (select list) reference. |
| source | The text to be moved to the target buffer or select list. Can be a literal enclosed in quotation marks or a reference to a buffer or select list. Multiple values can be specified as a comma-separated list. |

Description

The MV PROC command moves one or more values into a buffer or select list, or moves one or more values from one buffer (or select list) to another. Following the move operation, MV resets the buffer pointer to the beginning of the field(s) that were moved into target.

The three syntactical forms are equivalent. They are provided for compatibility with other MultiValue implementations.

The following source values are supported:

• "abc" — a literal string. A string may be delimited with single quotes ('abc'), double quotes ("abc"), or backslashes (\abc).
• I65 or X41 — a single character in either decimal encoding (Inn) or hexadecimal encoding (Xaa). For further details, refer to the T command.
• %3, #3, !3 — a numbered input buffer (%3), output buffer (#3) or select list (!3).
• #3,*2 — the asterisk specifies moving the next n fields from the current pointer position in the specified buffer. In this case, move the first two fields from output buffer 3. An asterisk without a number (for example, #3,* ) specifies moving all of the remaining fields from the current pointer position in the specified buffer.

Examples

The following example moves multiple string literals to the output buffer:

MV #2 "ABC","DEF","GHI"

See Also

• MVA PROC command
• MVD PROC command
• T PROC command
# MVA

Adds an element to a dynamic array.

## MVA target source

### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>target</td>
<td>A reference to a buffer or select list to be appended with the contents of source. A reference must begin with a %, !, &amp; or # character, followed by a number (%2), or followed by another of the four reference characters (%%).</td>
</tr>
<tr>
<td>source</td>
<td>The text to be appended to the target buffer or select list. source can be a literal or a reference to a buffer or select list.</td>
</tr>
</tbody>
</table>

### Description

The MVA PROC command appends a string to the contents of a buffer or select list. MVA adds the source string as an element to a dynamic array, separated by value mark (@VM) delimiters. It adds elements in collation order. If the contents of source matches an existing element in the dynamic array in target, no operation occurs.

### See Also

- MV PROC command
- MVD PROC command
MVD

Deletes an element from a buffer.

**MVD** target source

### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>target</strong></td>
<td>A reference to a buffer or select list from which the contents of source should be deleted. A reference must begin with a %, !, &amp;, or # character, followed by a number (%2), or followed by another of the four reference characters (%%).</td>
</tr>
<tr>
<td><strong>source</strong></td>
<td>The text to be deleted from the target buffer or select list. Can be a literal or a reference to a buffer or select list.</td>
</tr>
</tbody>
</table>

### Description

**MVD** deletes the source element from a dynamic array in target. It searches the dynamic array until it finds an exact match. It deletes the first matching element encountered and the associated value mark (@VM) delimiter. If the source value matches no existing element, no operation occurs.

### See Also

- MV PROC command
- MVA PROC command
O

Outputs to the terminal.

O [text [+]]

Arguments

| text | Optional — A string literal to output to the terminal. No string delimiters are required for a string literal. If text is omitted, O outputs a line return. |

Description

The O PROC command outputs a literal string to the user terminal. By default, the text string is followed by a line return. The optional + character suppresses the line return following text.

Commonly, the space between the command name and the text is omitted, as shown in the following example:

OInvalid Response - please re-enter.

See Also

- T PROC command
P, PH, PP, PW, PX

Executes a command from the primary output buffer.

P[options]

Arguments
None.

Description
The P PROC command processes a command in the primary output buffer (POB). It transfers the command to the command string for execution, then clears the POB.

In Caché and most emulations, the data stack and the secondary output buffer (SOB) use the same storage area. If the command requires an argument value, P takes this value from the secondary output buffer (SOB) and defines it as the data stack. When the PROC is executed (as opposed to linked to) it takes the value from the data stack, then clears the SOB and data stack.

P can take as a suffix the options P (display buffer contents, do not wait before executing), H (execute, hush output), W (display buffer contents then wait for user input before executing), or X (execute, upon completion exit the PROC), thus forming the PP, PH, PW, and PX commands. You can also specify multiple options in any order, such as PXW or PHP.

The PP and PW commands display the contents of the primary output buffer (POB) and secondary output buffer (SOB) before processing the command from the active output buffer. For the purpose of display, they convert @AM delimiter characters to spaces and remove leading blanks.

PP displays the contents of the output buffers, but does not wait or prompt for user input. It proceeds immediately to executing the displayed command.

PW displays the contents of the output buffers, then issues a terminal prompt, requiring the user to specify whether to process the displayed command or quit the PROC. The prompt acts on the first character typed; it does not require an Enter key. The prompt takes the following user-specified values:
• "Y" or the Enter key — Executes the displayed command, clears both output buffers and continues PROC execution.
• "N" or "X" — Exits the PROC and clears both output buffers.
• "S" — Skips (does not execute) the displayed command, clears both output buffers and continues PROC execution.

Emulation
In UniVerse and UniData, the data stack and the secondary output buffer (SOB) are separate storage areas. When the command requires an argument value, P takes this value from the secondary output buffer (SOB) and appends it to the existing data stack.

In most emulations, P strips leading blanks from a command prior to execution.

Following P command execution, the active select list is either retained or deleted, depending on the emulation. In Caché and the UniVerse, PICK, Prime, IN2, PIOPEN, and UniData emulations, the active select list is always retained. In jBASE, D3, R83, and REALITY emulations the active select list is discarded following the execution of a MultiValue Basic program.

In jBASE emulation, the PW command requires that you type a prompt letter then press the Enter key. (Just pressing the Enter key is the same as typing "Y" and pressing the Enter key.) Other emulations immediately process the prompt letter when you type it.

In Reality emulation, there is no PW command; the PP command behaves like the Caché MultiValue PW command.
In UniVerse emulation, the **PW "N"** prompt not only exits the PROC, but issues an abort, returning control to either the ON.ABORT verb or the MV shell command line. In other emulations, a "N" or "X" prompt stops execution but does not abort.

**See Also**

- O PROC command
**Q**

Quits PROC execution.

```
Q [text]
```

**Arguments**

| text | Optional — A message to display to the terminal when quitting PROC execution. |

**Description**

The `Q` PROC command quits PROC execution and returns to the command line interface.

**See Also**

- `U` PROC command
- `X` PROC command
RI

Reinitializes the input buffers.

```plaintext
RI [field]
```

**Arguments**

| field | Optional — Reinitializes (clears) all contents of the input buffer, beginning with this field. `field` is specified as a positive integer. |

**Description**

The **RI** PROC command clears both input buffers, resets the buffer pointers, and activates the primary input buffer (PIB). `RI` without an argument completely clears the input buffers. **RI field** clears the active input buffer of all contents beginning with (and including) the specified field. For example, **RI3** clears all of the buffer fields except 1 and 2.

Caché resets the input buffer pointer to the beginning of the input buffer following a **RInn** operation. UniVerse emulation (and other similar MultiValue emulations) also reset the buffer pointer to the beginning of the input buffer. D3 emulation (and other similar MultiValue emulations) reset the input buffer pointer to the end of the input buffer following a **RInn** operation.

**See Also**

- **RO PROC command**
**RO**

Reinitializes the output buffers.

| RO |

**Arguments**

None.

**Description**

The RO PROC command clears both output buffers, resets the buffer pointers, and activates the primary output buffer (POB). The active output buffer is referenced as #n, where n is an integer.

The RI command clears input buffers. The F-CLEAR command clears file buffers.

**See Also**

- RI PROC command
- F-CLEAR PROC command
RSUB

Returns to the GOSUB statement.

Arguments

None.

Description

The RSUB PROC command redirects execution to the next command after GOSUB. RSUB returns to the most recently invoked GOSUB. If no GOSUB has been invoked, no operation occurs.

See Also

- GOSUB PROC command
- IF PROC command
Sets a pointer in the active input buffer.

\[ \text{S [field]} \]

**Arguments**

| field | Optional — A field number. Specified as a number or as a reference to a buffer or a select list that contains the field number. |

**Description**

The \text{S} PROC command sets the primary input buffer (PIB) pointer to a specified field within the buffer. If \text{field} is larger than the number of fields in the PIB, \text{S} adds the specified number of empty elements.

**Emulation**

In Caché, UniVerse, and PICK, \text{S +field} is equivalent to \text{S field}. In all other emulations, \text{S +field} is equivalent to \text{S 1}.

REALITY emulation does not add empty elements when \text{field} is larger than the number of fields in the PIB.

**See Also**

- \text{T PROC command}
Activates the primary input buffer.

Arguments
None.

Description
The SP PROC command activates the primary input buffer as the current input buffer. This deactivates the secondary input buffer.

The SS PROC command activates the secondary input buffer, deactivating the primary input buffer. The RI command also activates the primary input buffer.

See Also
- RI PROC command
- SS PROC command
SS

Activates the secondary input buffer.

**Arguments**

None.

**Description**

The SS PROC command activates the secondary input buffer as the current input buffer. This deactivates the primary input buffer.

The SP and RI commands activate the primary input buffer, deactivating the secondary input buffer.

**See Also**

- RI PROC command
- SP PROC command
STOFF

Deactivates secondary output buffer (the stack).

**Arguments**

None.

**Description**

The STOFF PROC command deactivates the secondary output buffer (SOB), making the primary output buffer (POB) the active output buffer. By default, the secondary output buffer is inactive at the beginning of a PROC. The secondary output buffer is activated by the STON command.

**See Also**

- RO PROC command
- STON PROC command
STON

Activates secondary output buffer (the stack).

Arguments

None.

Description

The STON PROC command activates the secondary output buffer (SOB), which functions as the command stack for the PROC. By default, the secondary output buffer is inactive at the beginning of a PROC. The secondary output buffer is deactivated by the STOFF command or the RO command.

See Also

- **RO PROC command**
- **STOFF PROC command**
Displays to Terminal.

T [text] [+]  
T Inn  
T Xaa  
T code  
T (col,row)  
T (-n)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td><em>Optional</em> — A string to be displayed on the terminal screen. <em>text</em> can be a text string enclosed, or a reference to a buffer or a select list that contains the text string.</td>
</tr>
<tr>
<td>Inn</td>
<td>A single character to be displayed on the terminal screen, specified by its decimal encoding. <em>nn</em> is a two-digit number. For example, I65 displays the character A.</td>
</tr>
<tr>
<td>Xaa</td>
<td>A single character to be displayed on the terminal screen, specified by its hexadecimal encoding. <em>aa</em> is a two-digit hexadecimal number. For example, X41 displays the character A.</td>
</tr>
<tr>
<td>code</td>
<td>A display control operation. Specified as a single letter code. B = ring the bell; C = clear the screen; Sn = insert n spaces; U = move cursor up one line.</td>
</tr>
<tr>
<td>(col,row)</td>
<td>A move cursor operation. Specified as positive integers for <em>col</em> (column) and <em>row</em> position. To specify just column position: <em>(col)</em>. To specify just row position: <em>(row)</em>.</td>
</tr>
<tr>
<td>(-n)</td>
<td>A cursor control operation. Specified as a negative integer code number enclosed in parentheses.</td>
</tr>
</tbody>
</table>

Description

The T PROC command controls terminal display. It displays a specified text and performs other terminal display operations.

A *text* string may be delimited with single quotes (‘abc’), double quotes (“abc”), or backslashes (\abc\). By default, the *text* string is followed by an automatic line return. If *text* is omitted, T just issues a line return. If *text* is specified, the optional + character suppresses the line return following *text*.

You can specify multiple terminal display arguments as a comma-separated list. Arguments are executed in left-to-right order. You can introduce a line break into comma-separated argument list following a comma. No automatic line return is performed between items in a T command argument list. If there are multiple arguments, the optional + character can only appear following the final argument.

The T command ignores any code or code component that it cannot parse, and proceeds to the next comma-separated argument. For example, T C,"hello world" and T CLEAR,"hello world" perform the same operations; the LEAR letters are ignored.

See Also

- P PROC command
TR

Activates trace processing for PROC debugging.

| TR  | ON | OFF |

Arguments
None

Description
The TR PROC command is used to active debug tracing.

See Also
- T PROC command
U

Branches to user exit MVBasic routine.

**U hex**

<table>
<thead>
<tr>
<th><strong>Arguments</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>hex</strong></td>
</tr>
</tbody>
</table>

**Description**

The **U** PROC command exits PROC execution and branches execution to a catalogued MVBasic program. This command is commonly used to perform exception handling. Commonly, the space between the command name and `hex` is omitted. For example, **U31AD**.

**See Also**

- Q PROC command
- X PROC command
**X**

Exits on error and returns to the calling environment.

```
X [text] [+]
```

**Arguments**

<table>
<thead>
<tr>
<th>text</th>
<th>Optional — A message to display to the terminal when exiting a PROC.</th>
</tr>
</thead>
</table>

**Description**

The X PROC command is used to exit a PROC when an error occurs. It can optionally display an error message when exiting. By default, the `text` string is followed by a line return. The optional + character suppresses the line return following `text`.

**Examples**

The following examples show the X command exiting the PROC and issuing an error message when an open operation fails. Note that the space between the command name and `text` is optional.

```text
POQ
F-OPEN 7 VOC
X Failed Open
F-READ 7 MyData

POQ
F-OPEN 7 VOC
X:Failed Open
F-READ 7 MyData
```

**See Also**

- Q PROC command
- U PROC command