

PROGRESS®

POCKET PROGRESS

Copyright© 1990 Progress Software Corporation
617-275-4500

PROGRESS is copyrighted and all rights are reserved by Progress Software Corporation. This manual is copyrighted and all rights are reserved. This document may not, in whole or part, be copied, photocopied, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from Progress Software Corporation.

The information in this manual is subject to change without notice and should not be construed as a commitment by Progress Software Corporation. Progress Software Corporation assumes no responsibility for any errors that may appear in this document.

PROGRESS® is a registered trademark of Progress Software Corporation.

Printed in U.S.A.

December 1990

Product names are trademarks of their respective manufacturers.

CONTENTS

| | |
|--|-----------|
| STARTING AND RUNNING PROGRESS | 1 |
| STARTUP OPTIONS | 13 |
| ENVIRONMENT VARIABLES | 42 |
| THE PROGRESS KEYBOARD | 49 |
| KEYCODES AND KEY LABELS | 59 |
| THE PROGRESS LANGUAGE | 67 |
| PROGRESS SYNTAX RULES | 68 |
| PROGRESS SYNTAX SUMMARY | 70 |

| | |
|--|------------|
| OPERATOR PRECEDENCE TABLE | 182 |
| DATA FORMATS | 183 |
| NUMERIC FORMAT | 185 |
| TIME FORMAT (Used with the STRING Function) | 185 |
| DATA HANDLING STATEMENTS AND DATA MOVEMENT | 187 |
| STATEMENTS AS BUILDING BLOCKS | 188 |
| BLOCK PROPERTIES | 189 |
| PROGRESS FUNCTION AND CONTROL KEYS | 209 |
| MASTER INDEX | |
| PROGRESS FUNCTION AND CONTROL KEYS | |

STARTING AND RUNNING PROGRESS

In the following commands, *database-name* is the name of the database you are using. On DOS and OS/2, the database-name can be a maximum of eight characters, with no file extension; on UNIX, the database-name can be a maximum of eleven characters. Braces { } indicate an item is required. Brackets [] indicate an item is optional. See the table on pages 13-38 for a list of startup options you can use in conjunction with these commands.

| Operating System | Command Syntax | Command Action |
|------------------|---|-------------------------------|
| UNIX, DOS & OS/2 | prodb <i>database-name</i> empty | Creates a new empty database. |
| VMS | PROGRESS/CREATE <i>database-name</i> empty | |
| BTOS/CTOS | PROGRESS Create Database New Database Name <i>database-name</i> Copy From Database Name empty | |

STARTING AND RUNNING PROGRESS

| Operating System | Command Syntax | Command Action |
|------------------|---|--|
| UNIX, DOS & OS/2 | prodb <i>database-name</i> demo | Creates a copy of the PROGRESS demonstration database. |
| VMS | PROGRESS/CREATE <i>database-name</i> demo | |
| BTOS/CTOS | PROGRESS Create Database New Database Name <i>database-name</i> Copy From Database Name demo | |
| UNIX, DOS & OS/2 | prodb <i>database-name</i> <i>parent-database-name</i> | Creates a copy of an existing database (<i>parent-database</i>). |
| VMS | PROGRESS/CREATE <i>database-name</i> <i>parent-database-name</i> | |
| BTOS/CTOS | PROGRESS Create Database New Database Name <i>database-name</i> Copy From Database Name <i>parent-database-name</i> | |

STARTING AND RUNNING PROGRESS

| Operating System | Command Syntax | Command Action |
|------------------|--------------------------|--|
| UNIX, DOS & OS/2 | pro [<i>options</i>] | Start PROGRESS with no database connected. |
| VMS | PROGRESS/ <i>options</i> | |
| BTOS/CTOS | PROGRESS [Options] | |

STARTING AND RUNNING PROGRESS

| Operating System | Command Syntax | Command Action |
|------------------|---|------------------------------|
| UNIX, DOS & OS/2 | pro <i>database-name</i> [<i>options</i>] | Starts single-user PROGRESS. |
| VMS | PROGRESS/OPTIONS <i>database-name</i> | |
| BTOS/CTOS | PROGRESS 4GL [Options] -1 <i>database-name</i> | |

STARTING AND RUNNING PROGRESS

| Operating System | Command Syntax | Command Action |
|------------------|--|---|
| UNIX, DOS & OS/2 | <code>bpro database-name -p procedure-name [options] > error-file</code> | Runs single-user PROGRESS in batch or background mode. |
| VMS | <code>PROGRESS/BATCH = batch-queue/JOB_NAME = job name /STARTUP = procedure-name database-name</code> | |
| BTOS/CTOS | <code>PROGRESS 4GL [Start-Up Procedure -pf] procedure-name ⋮ [Batch?] yes ⋮ [Options] database-name</code> | |

STARTING AND RUNNING PROGRESS

| Operating System | Command Syntax | Command Action |
|------------------|--|--------------------------------------|
| UNIX, DOS & OS/2 | proserve <i>database-name</i> [<i>options</i>] | Starts a multi-user database server. |
| VMS | PROGRESS/MULTI_USER = START_SERVER/ <i>qualifiers database-name</i> | |
| BTOS/CTOS | PROGRESS Server Database Name <i>database-name</i> [Options] | |

STARTING AND RUNNING PROGRESS

| Operating System | Command Syntax | Command Action |
|------------------|---|---|
| UNIX, DOS & OS/2 | <code>mpro database-name [options]</code> | Starts a multi-user PROGRESS session (a server must have been previously started for the database). |
| VMS | <code>PROGRESS/MUTLI_USER = LOGIN/qualifiers database-name</code> | |
| BTOS/CTOS | <code>PROGRESS 4GL (to run the resident version)</code> <code>[Options] database-name</code> | |

STARTING AND RUNNING PROGRESS

| Operating System | Command Syntax | Command Action |
|------------------|---|---|
| UNIX, DOS & OS/2 | <code>mbpro database-name -p procedure-name</code> <code>[options] >error-file</code> | Starts a multi-user PROGRESS batch or background session. |
| VMS | <code>PROGRESS/MULTI_USER = LOGIN/</code> <code>BATCH = batch-queue/JOB_NAME = job-name</code> <code>/STARTUP = procedure-name</code> <code>database-name</code> | |
| BTOS/CTOS | <code>PROGRESS 4GL</code> <code>[Start-Up Procedure -p] procedure-name</code> <code>[Batch -b] Yes</code> <code>[Options] database-name</code> | |

STARTING AND RUNNING PROGRESS

| Operating System | Command Syntax | Command Action |
|------------------|--|-------------------------------|
| UNIX, DOS & OS/2 | <code>proshut <i>database-name</i></code> | Shuts down a database server. |
| VMS | <code>PROGRESS/MULTI_USER = SHUTDOWN <i>database-name</i></code> | |
| BTOS/CTOS | <p>PROGRESS Shutdown Server</p> <p style="padding-left: 40px;">Database Name <i>database-name</i></p> <p style="padding-left: 40px;">[Server Name]</p> <p style="padding-left: 40px;">[IPC Block Size]</p> <p style="padding-left: 40px;">[Options]</p> | |

STARTING AND RUNNING PROGRESS

| Operating System | Command Syntax | Command Action |
|-------------------------|--|--|
| UNIX, DOS & OS/2 VMS | prolog <i>database-name</i> PROGRESS/PURGE_LOG <i>database-name</i> | Removes all but the most recent entries in the log (.lg) file. |
| BTOS/CTOS | PROGRESS Log Maintenance Database Name <i>database-name</i> | |
| UNIX, DOS & OS/2 VMS | prodel <i>database-name</i> PROGRESS/DELETE <i>database-name</i> | Deletes a database. |
| BTOS/CTOS | PROGRESS Delete Database Database Name <i>database-name</i> | |

STARTING AND RUNNING PROGRESS

| Operating System | Command Syntax | Command Action |
|--------------------------------------|---|---|
| UNIX, DOS & OS/2 VMS BTOS/CTOS | <pre> quoter <i>input-file</i> ><i>output-file</i> PROGRESS/TOOLS=QUOTER/OUTPUT=<i>output-file input-file</i> Quoter Input File <i>input-file</i> [Options] ><i>output-file</i> </pre> | Reads input from a file, places quotes (") at the beginning and end of each line, and replaces each quote in the data with two quote characters. |
| UNIX, DOS & OS/2 VMS BTOS/CTOS | <pre> quoter <i>input-file</i> -d <i>delimiter-character</i> ><i>output-file</i> PROGRESS/TOOLS=QUOTER/Delimiter="<i>character</i>"/OUPUT =<i>output-file input file</i> Quoter Input File <i>input-file</i> [Options] -d "<i>character</i>" ><i>output-file</i> </pre> | Reads input from a file, separates each line into fields based on a delimiter character, and puts the fields into a format for use with the INPUT FROM or INPUT THROUGH statements. |

STARTING AND RUNNING PROGRESS

| Operating System | Command Syntax | Command Action |
|------------------|--|--|
| UNIX, DOS & OS/2 | $\text{quoter } \textit{input-file} \text{ -c } \left\{ \textit{col-col} [, \textit{col-col}_n] \dots \right\}$ $\text{ > } \textit{output file}$ | Reads input from a file, separates each line into fields based on specified starting and ending column numbers, and puts the fields into a format for use with the INPUT FROM or INPUT THROUGH statements. |
| VMS | $\text{PROGRESS/TOOLS} = \text{QUOTER/COLUMN} = \textit{"col-col [, col-col_n]"} \\ \dots \textit{"/OUTPUT} = \textit{output-file input-file}$ | |
| BTOS/CTOS | Quoter Input File $\textit{input-file}$ [Options] $\text{-c } \textit{startcol} \text{ - } \textit{stopcol} \dots \text{ > } \textit{output-file}$ | |

STARTUP OPTIONS

The symbols you use for the option syntax are case sensitive. You must type them exactly as shown.

| Option | UNIX/ DOS/ OS/2 | VMS | BTOS/CTOS | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default |
|------------------------------------|-----------------------|--|-------------------------------------|---------------------------|--------------|--------------|----------------------------|-----------------------|
| | | | | | | | | |
| Single user ⁷ | -1 | | [options] -1 | M | | | | |
| After image file name ⁷ | -a <i>ai-file</i> | /AFTER_IMAGE= <i>after-image-file</i> | [After Image File] <i>ai-file</i> | P, S proutil rfutil | | | | |
| No tabs in editor out- put | -A | /NOTAB | [Detab Editor Output ?] yes | P, M | | | | |

1. Relevant only for UNIX and VMS systems that use shared memory.
2. UNIX only. 2a. UNIX X Windows only.
3. Automatically supplied with bpro and apro commands
4. The maximum is 63KB on INTEL 8086 and 80286 processors.

5. The practical maximum is considerably lower and is system dependent.

6. Internal use only.

7. Can be used with CONNECT.

8. The practical maximum is system-dependent but is always at least 10.

* (M) Multi-user startup commands

* (P) Single-user startup commands

* (S) Multi-user server/broker startup commands

| Option | UNIX/ DOS/ OS/2 | | VMS | BTOS/CTOS | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default |
|------------------|-----------------------|--|---|-------------|--------|--------------|--------------|----------------------------|-----------------------|
| | | | | | | | | | |
| Batch 7 | -b 3 | | /BATCH= <i>batch-queue</i> /NOINTERACTIVE /JOB_NAME= <i>job-name</i> /STARTUP= <i>procedure-name</i> | [batch] yes | P, M | | | | |
| Border color | -bd <i>color</i> 2a | | | | P, M | | | | |
| Background color | -bg <i>color</i> 2a | | | | P, M | | | | |

- 1. Relevant only for UNIX and VMS systems that use shared memory.
- 2. UNIX only. 2a. UNIX X Windows only.
- 3. Automatically supplied with bpro and mpro commands
- 4. The maximum is 63KB on systems designed for the INTEL 8086 and 80286 processors and on BTOS/CTOS machines.

- 5. The practical maximum is considerably lower and is system dependent.
- 6. Internal use only.
- 7. Can be used with CONNECT .
- 8. The practical max is system dependent but is always at least 10.

- * (M) Multi-user startup commands
- * (P) Single-user startup commands
- * (S) Multi-user server/broker startup commands

STARTUP OPTIONS

The symbols you use for the option syntax are case sensitive. You must type them exactly as shown.

| Option | UNIX/ DOS/ OS/2 | | VMS | | BTOS/CTOS | | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default |
|----------------------------|---------------------------------|--|---------------------------|--|----------------------------------|--|---------------------------|--------------|--------------|----------------------------|-----------------------|
| | | | | | | | | | | | |
| No-kill users | -bn | | /PROCEED_IF_NO_USERS | | [options] -bn | | proshut | | | | |
| Border width | -bw <i>number</i> ^{2a} | | | | | | P, M | | | | |
| Kill users | -by | | /KILL_USERS | | [options] -by | | proshut | | | | |
| Blocks in database buffers | 7 -B <i>n</i> | | /BUFFERS = <i>integer</i> | | [Blocks in DB Buffers] <i>n</i> | | P, S proutil rfutil | 32000 | 10 | 20 | (8 * users) |

1. Relevant only for UNIX and VMS systems that use shared memory.
2. UNIX only. 2a. UNIX X Windows only.
3. Automatically supplied with bpro and mpro commands
4. The maximum is 63KB on INTEL 8086 and 80286 processors.

5. The practical maximum is considerably lower and is system dependent.
6. Internal use only.
7. Can be used with CONNECT.
8. The practical maximum is system-dependent but is always at least 10.

- * (M) Multi-user startup commands
- * (P) Single-user startup commands
- * (S) Multi-user server/broker startup commands

| Option | UNIX/ DOS/ OS/2 | VMS | BTOS/CTOS | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default |
|--------------------------------|-----------------------|-------------------------------------|--------------------------------|---------------|--------------|--------------|----------------------------|-----------------------|
| | | | | | | | | |
| Index cursors | -c <i>n</i> | /INDEX_CURSORS = <i>integer</i> | [Index Cursors] <i>n</i> | P, S | 1000 | 10 | 20 | (4 * users) |
| Configuration File | -cfg <i>filename</i> | /CONFIG_FILE = <i>filename</i> | [Options] -cfg <i>filename</i> | P, S, M | | | | |
| Multi-user client ⁷ | -cl | -cl (<i>parameter file only</i>) | | M, proshut | | | | |
| Communications Parameter File | -cp <i>params</i> | /COMM_PARMFILE = <i>filename</i> | | P, S, M | | | | |

1. Relevant only for UNIX and VMS systems that use shared memory.
2. UNIX only. 2a. UNIX X Windows only.
3. Automatically supplied with `bpro` and `mpro` commands
4. The maximum is 63KB on systems designed for the INTEL 8086 and 80286 processors and on BTOS/CTOS machines.

5. The practical maximum is considerably lower and is system dependent.
6. Internal use only.
7. Can be used with `CONNECT`.
8. The practical max is system dependent but is always at least 10.

- * (M) Multi-user startup commands
- * (P) Single-user startup commands
- * (S) Multi-user server/broker startup commands

STARTUP OPTIONS

The symbols you use for the option syntax are case sensitive. You must type them exactly as shown.

| Option | UNIX/ DOS/ OS/2 | VMS | BTOS/CTOS | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default | |
|--|-----------------------|--------------------|-------------------------------|---|-----------------------------|--------------|----------------------------|-----------------------|-----|
| | | | | | | | | | |
| Cursor size | 7 | -cs <i>n</i> | /CURSOR_SIZE = <i>integer</i> | [Options] -cs | P, S, proutil, rfutil | 256 | 1 | 6 | 6 |
| Option on proutil and rfutil commands | | -C | /UTILITIES = <i>option</i> | PROGRESS Utilities Database Name <i>databasename</i> Utility Name <i>utility-name</i> [Options] <i>options</i> | proutil, rfutil | | | | |
| Date format | | -d <i>dateform</i> | /DATE_FORMAT = <i>string</i> | [Date Format] <i>dateform</i> | P, M | | | mdy | mdy |

1. Relevant only for UNIX and VMS systems that use shared memory.
 2. UNIX only. 2a. UNIX X Windows only.
 3. Automatically supplied with `bpro` and `mpro` commands
 4. The maximum is 63KB on INTEL 8086 and 80286 processors.
 5. The practical maximum is considerably lower and is system dependent.
 6. Internal use only.
 7. Can be used with `CONNECT`.
 8. The practical maximum is system-dependent but is always at least 10.
- * (M) Multi-user startup commands
 - * (P) Single-user startup commands
 - * (S) Multi-user server/broker startup commands

| Option | UNIX/ DOS/ OS/2 | VMS | BTOS/CTOS | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default |
|-------------------------------------|--|--|--------------------------------------|---------|--------------|--------------|----------------------------|-----------------------|
| | | | | | | | | |
| Direct access ⁷ | -da | -da (<i>parameter file only</i>) | not applicable | M | | | | |
| Physical database name ⁷ | -db <i>physical- dbname</i> | -db <i>physical-dbname (parameter file only)</i> | [Options] -db <i>physical-dbname</i> | P, S, M | | | | |
| Display host | -display ^{2a} <i>hostname: server:screen</i> | | | P, M | | | | |

- 1. Relevant only for UNIX and VMS systems that use shared memory.
- 2. UNIX only. 2a. UNIX X Windows only.
- 3. Automatically supplied with bpro and mpro commands
- 4. The maximum is 63KB on systems designed for the INTEL 8086 and 80286 processors and on BTOS/CTOS machines.

- 5. The practical maximum is considerably lower and is system dependent.
- 6. Internal use only.
- 7. Can be used with CONNECT .
- 8. The practical max is system dependent but is always at least 10.

- * (M) Multi-user startup commands
- * (P) Single-user startup commands
- * (S) Multi-user server/broker startup commands

STARTUP OPTIONS

The symbols you use for the option syntax are case sensitive. You must type them exactly as shown.

| Option | UNIX/ DOS/ OS/2 | | | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default |
|--------------------------|-----------------------|---|------------------------------|---------|---------------------------------|--------------|----------------------------|-----------------------|
| | VMS | BTOS/CTOS | | | | | | |
| Database type | 7 -dt <i>dbtype</i> | -dt <i>dbtype</i> (parameter file only) | not applicable | P, S, M | | | PROG- RESS | PROGRESS |
| Directory size | -D <i>n</i> | /COMPILED_FILE_ DIRECTORY = <i>integer</i> | [Directory Size] <i>n</i> | P, M | 500 entries | 5 entries | 36 entries | 36 entries |
| Edit buffer size (in KB) | -e <i>KB</i> | /EDIT_BUFFER = <i>integer</i> | [Edit Buffer Size] <i>n</i> | P, M | (4GB) ⁴ ⁴ | 2 | 32 | 32 |

1. Relevant only for UNIX and VMS systems that use shared memory.
2. UNIX only. 2a. UNIX X Windows only.
3. Automatically supplied with bpro and mpro commands
4. The maximum is 63KB on INTEL 8086 and 80286 processors.

5. The practical maximum is considerably lower and is system dependent.

6. Internal use only.

7. Can be used with CONNECT.

8. The practical maximum is system-dependent but is always at least 10.

* (M) Multi-user startup commands

* (P) Single-user startup commands

* (S) Multi-user server/broker startup commands

| Option | UNIX/ DOS/ OS/2 | VMS | BTOS/CTOS | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default |
|-----------------------------------|--|--|--------------------------------|--------|--------------|--------------|----------------------------|-----------------------|
| | | | | | | | | |
| Expanded memory size ⁷ | -ems (DOS only) | not applicable | not applicable | P, M | | | | |
| European numeric format | -E | /NuMERIC FORMAT= {AMERICAN} {EUROPEAN} | [European Number Format?] yes | P, M | | | | |
| Font | -fn <i>fontname</i> ^{2a} or -font <i>fontname</i> | | | P, M | | | | |

1. Relevant only for UNIX and VMS systems that use shared memory.
2. UNIX only. 2a. UNIX X Windows only.
3. Automatically supplied with bpro and mpro commands
4. The maximum is 63KB on systems designed for the INTEL 8086 and 80286 processors and on BTOS/CTOS machines.

5. The practical maximum is considerably lower and is system dependent.

6. Internal use only.

7. Can be used with CONNECT.

8. The practical max is system dependent but is always at least 10.

* (M) Multi-user startup commands

* (P) Single-user startup commands

* (S) Multi-user server/broker startup commands

STARTUP OPTIONS

The symbols you use for the option syntax are case sensitive. You must type them exactly as shown.

| Option | UNIX/ DOS/ OS/2 | VMS | BTOS/CTOS | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default |
|-------------------------------------|--|--------------------------------|------------------------------------|-----------------------------|--------------|--------------|----------------------------|-----------------------|
| | | | | | | | | |
| Foreground color | -fg <i>color</i> ^{2a} or -foreground <i>color</i> | | | P, M | | | | |
| Force access | -F | /ACCESS = {NORMAL} {FORCED} | [options] -F | P, S proshut | | | | |
| Before image file name ⁷ | -g <i>bi-file</i> | /BEFORE_IMAGE = <i>bi-file</i> | [Before Image File] <i>bi-file</i> | proutil, rfutil, P, S | | | | |

1. Relevant only for UNIX and VMS systems that use shared memory.
2. UNIX only. 2a. UNIX X Windows only.
3. Automatically supplied with `bpro` and `mpro` commands
4. The maximum is 63KB on INTEL 8086 and 80286 processors.

5. The practical maximum is considerably lower and is system dependent.
6. Internal use only.
7. Can be used with CONNECT.
8. The practical maximum is system-dependent but is always at least 10.

- * (M) Multi-user startup commands
- * (P) Single-user startup commands
- * (S) Multi-user server/broker startup commands

| Option | UNIX/ DOS/ OS/2 | | VMS | BTOS/CTOS | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default |
|--------------------------------|---|----------------------------------|-----|-------------------------|-------------------|--------------|--------------|----------------------------|-----------------------|
| | | | | | | | | | |
| Geometry | -geometry ^{2a} widthxheight +xoff+yoff | | | | P, M | | | | |
| Before image truncate interval | -G n | /ROLL_FORWARD_INTERVAL = integer | | [options] -G n | proutil rfutil | | | | |
| Number of databases | -h n | /MAXDATABASES = n | | [Number of Databases] n | P, S, M | 240 | 1 | 5 | 5 |

- 1. Relevant only for UNIX and VMS systems that use shared memory.
- 2. UNIX only. 2a. UNIX X Windows only.
- 3. Automatically supplied with `bpro` and `mpro` commands
- 4. The maximum is 63KB on systems designed for the INTEL 8086 and 80286 processors and on BTOS/CTOS machines.

- 5. The practical maximum is considerably lower and is system dependent.
- 6. Internal use only.
- 7. Can be used with `CONNECT`.
- 8. The practical max is system dependent but is always at least 10.

- * (M) Multi-user startup commands
- * (P) Single-user startup commands
- * (S) Multi-user server/broker startup commands

STARTUP OPTIONS

The symbols you use for the option syntax are case sensitive. You must type them exactly as shown.

| Option | UNIX/ DOS/ OS/2 | VMS | BTOS/CTOS | Usage * | Max Value | Min Value | Single- user Default | Multi-user Default |
|----------------------------------|----------------------------|--------------------------|----------------|-----------------|--------------|--------------|----------------------------|-----------------------|
| | | | | | | | | |
| Heap size | -hs <i>KB</i> ² | | not applicable | P, S | 2000 | 1 | Sys. de- pendent | System Dependent |
| Host name ⁷ | -H <i>host-name</i> | /HOST = <i>host-name</i> | not applicable | M | | | | |
| No crash protection ⁷ | -i | /NORECOVERY | [Options] -i | P, S, rfutil | | | | |

1. Relevant only for UNIX and VMS systems that use shared memory.
2. UNIX only. 2a. UNIX X Windows only.
3. Automatically supplied with bpro and mpro commands
4. The maximum is 63KB on INTEL 8086 and 80286 processors.

5. The practical maximum is considerably lower and is system dependent.

6. Internal use only.

7. Can be used with CONNECT.

8. The practical maximum is system-dependent but is always at least 10.

* (M) Multi-user startup commands

* (P) Single-user startup commands

* (S) Multi-user server/broker startup commands

| Option | UNIX/ DOS/ OS/2 | VMS | BTOS/CTOS | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default |
|-----------------------|-------------------------------------|---|-----------|--------|--------------------|--------------|----------------------------|-----------------------|
| | | | | | | | | |
| Icon | -icon <i>filename</i> ^{2a} | | | P, M | | | | |
| Iconic | -iconic ^{2a} | | | P, M | | | | |
| Total private buffers | -I <i>buffers</i> | /TOTAL_PRIVATE_ BUFFERS = <i>integer</i> | | S | 32000 ⁵ | 0 | 0 | 2 * users |

1. Relevant only for UNIX and VMS systems that use shared memory.
2. UNIX only. 2a. UNIX X Windows only.
3. Automatically supplied with bpro and mpro commands
4. The maximum is 63KB on systems designed for the INTEL 8086 and 80286 processors and on BTOS/CTOS machines.

5. The practical maximum is considerably lower and is system dependent.
6. Internal use only.
7. Can be used with CONNECT .
8. The practical max is system dependent but is always at least 10.

- * (M) Multi-user startup commands
- * (P) Single-user startup commands
- * (S) Multi-user server/broker startup commands

STARTUP OPTIONS

The symbols you use for the option syntax are case sensitive. You must type them exactly as shown.

| Option | UNIX/ DOS/ OS/2 | | VMS | BTOS/CTOS | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default |
|------------------------------|-------------------------------|--|-----|--|--------|------------------|--------------|----------------------------|-----------------------|
| | | | | | | | | | |
| Keyword forget | -k <i>filename</i> | /KEYWORD_FORGET = <i>filename</i> | | [Keyword Forget List] <i>filename</i> | P, M | | | | |
| Local buffer size (in KB) | 7 -l <i>KB</i> | /LOCAL_BUFFER_SIZE= <i>integer</i> | | [Local Buffer Size] <i>n</i> | P, M | 4GB ⁴ | 1 | 10 | 10 |
| 7 Logical database name | -ld <i>logical dbname</i> | /LOGICAL_DBNAME = <i>logical-dbname</i> | | [Options] -ld <i>logical-dbname</i> | P, M | | | | |

1. Relevant only for UNIX and VMS systems that use shared memory.
 2. UNIX only. 2a. UNIX X Windows only.
 3. Automatically supplied with bpro and mpro commands
 4. The maximum is 63KB on INTEL 8086 and 80286 processors.
 5. The practical maximum is considerably lower and is system dependent.
 6. Internal use only.
 7. Can be used with CONNECT.
 8. The practical maximum is system-dependent but is always at least 10.
- * (M) Multi-user startup commands
 - * (P) Single-user startup commands
 - * (S) Multi-user server/broker startup commands

| Option | UNIX/ DOS/ OS/2 | VMS | BTOS/CTOS | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default |
|-------------------------------|----------------------------|------------------------------|----------------------------------|---------|--------------|--------------|----------------------------|-----------------------|
| | | | | | | | | |
| Leave memory | -lm <i>n</i> (DOS only) | not applicable | not applicable | P, S, M | 200 | 0 | 0 | 0 |
| Locking table entries | -L <i>n</i> | /LOCK_TABLE = <i>integer</i> | [Locking Table Entries] <i>n</i> | S | > = 2000 | 32 | | 500 |
| Spawned server ^{1 6} | -m1 ² | /SERVER_TYPE = AUTO | | S | | | | |
| Manual server ¹ | -m2 ² | /SERVER_TYPE = MANUAL | | S | | | | |

1. Relevant only for UNIX and VMS systems that use shared memory.

2. UNIX only. 2a. UNIX X Windows only.

3. Automatically supplied with bpro and mpro commands

4. The maximum is 63KB on systems designed for the INTEL 8086 and 80286 processors and on BTOS/CTOS machines.

5. The practical maximum is considerably lower and is system dependent.

6. Internal use only.

7. Can be used with CONNECT.

8. The practical max is system dependent but is always at least 10.

* (M) Multi-user startup commands

* (P) Single-user startup commands

* (S) Multi-user server/broker startup commands

STARTUP OPTIONS

The symbols you use for the option syntax are case sensitive. You must type them exactly as shown.

| Option | UNIX/ DOS/ OS/2 | VMS | BTOS/CTOS | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default |
|--------------------------------------|---------------------------------|--|-----------|--------|--------------|--------------|----------------------------|-----------------------|
| | | | | | | | | |
| Secondary login broker ¹ | -m ³ ² | /SERVER_TYPE = LOGIN | | S | | | | |
| Max clients per server ¹ | -Ma <i>clients</i> ² | /MAXCLIENTS = <i>integer</i> | | S | 2048 | 1 | | users/servers |
| Suppress .bi file write ⁷ | -Mf <i>seconds</i> | /TRANSACTION_DELAY = <i>seconds</i> | | P, S | 32768 | 0 | 0 | 0 |

1. Relevant only for UNIX and VMS systems that use shared memory.
2. UNIX only. 2a. UNIX X Windows only.
3. Automatically supplied with bpro and mpro commands
4. The maximum is 63KB on INTEL 8086 and 80286 processors.

5. The practical maximum is considerably lower and is system dependent.
6. Internal use only.
7. Can be used with CONNECT.
8. The practical maximum is system-dependent but is always at least 10.

- * (M) Multi-user startup commands
- * (P) Single-user startup commands
- * (S) Multi-user server/broker startup commands

| Option | UNIX/ DOS/ OS/2 | VMS | BTOS/CTOS | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default |
|--------------------|-----------------------|-------------------------------|----------------------------|--------|--------------|--------------|----------------------------|-----------------------|
| | | | | | | | | |
| Put screen refresh | -psc 2a | | | P, M | | | | |
| Password 7 | -P <i>password</i> | /PASSWORD = " <i>string</i> " | [Password] <i>password</i> | P, M | | | | |
| Quick request | -q | /QUICK_REQUEST | [Quick Request ?] yes | P, M | | | | |
| ANSI SQL | -Q | /ANSI_SQL | [ANSI SQL ?] yes | P, M,S | | | | |

- 1. Relevant only for UNIX and VMS systems that use shared memory.
- 2. UNIX only. 2a. UNIX X Windows only.
- 3. Automatically supplied with bpro and mpro commands
- 4. The maximum is 63KB on systems designed for the INTEL 8086 and 80286 processors and on BTOS/CTOS machines.

- 5. The practical maximum is considerably lower and is system dependent.
- 6. Internal use only.
- 7. Can be used with CONNECT .
- 8. The practical max is system dependent but is always at least 10.

- * (M) Multi-user startup commands
- * (P) Single-user startup commands
- * (S) Multi-user server/broker startup commands

STARTUP OPTIONS

The symbols you use for the option syntax are case sensitive. You must type them exactly as shown.

| Option | UNIX/ DOS/ OS/2 | VMS | BTOS/CTOS | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default |
|----------------------------|-----------------------------|----------------|----------------|----------------|--------------|--------------|----------------------------|-----------------------|
| | | | | | | | | |
| Buffered I/O ⁷ | -r ² | not applicable | not applicable | P, S rfutil | | | Raw I/O | Raw I/O |
| Return fault table | -rft <i>n</i> (DOS only) | not applicable | not applicable | P, M | 1000 | 25 | 100 | 100 |
| Encrypted Compiler Mode | -rx | /XCOMPILER | [Options] -rx | P, S, M | | | | |

1. Relevant only for UNIX and VMS systems that use shared memory.
2. UNIX only. 2a. UNIX X Windows only.
3. Automatically supplied with bpro and mpro commands
4. The maximum is 63KB on INTEL 8086 and 80286 processors.

5. The practical maximum is considerably lower and is system dependent.

6. Internal use only.

7. Can be used with CONNECT.

8. The practical maximum is system-dependent but is always at least 10.

* (M) Multi-user startup commands

* (P) Single-user startup commands

* (S) Multi-user server/broker startup commands

| Option | UNIX/ DOS/ OS/2 | | | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default |
|--------------------------|--------------------------|-------------------------------|----------------------------|---------|--------------|--------------|----------------------------|-----------------------|
| | VMS | BTOS/CTOS | | | | | | |
| Raw I/O ⁷ | -R ² | not applicable | not applicable | P, S | | | Raw I/O | Raw I/O |
| Read only ⁷ | -RO | /READONLY | [Options] -RO | P | | | | |
| Stack size (in KB) | -s <i>n</i> | /STACK = <i>integer</i> | [Stack Size] <i>n</i> | P, S, M | 31 | 2 | 12 | 12 |
| Server name ⁷ | -S <i>server-name</i> | /SERVICE = <i>server-name</i> | [Server Name] <i>name</i> | P, M | | | | |

1. Relevant only for UNIX and VMS systems that use shared memory.
2. UNIX only. 2a. UNIX X Windows only.
3. Automatically supplied with `bpro` and `mpro` commands
4. The maximum is 63KB on systems designed for the INTEL 8086 and 80286 processors and on BTOS/CTOS machines.

5. The practical maximum is considerably lower and is system dependent.
6. Internal use only.
7. Can be used with `CONNECT`.
8. The practical max is system dependent but is always at least 10.

- * (M) Multi-user startup commands
- * (P) Single-user startup commands
- * (S) Multi-user server/broker startup commands

STARTUP OPTIONS

The symbols you use for the option syntax are case sensitive. You must type them exactly as shown.

| Option | UNIX/ DOS/ OS/2 | | VMS | | BTOS/CTOS | | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default |
|---------------------|---------------------------------------|--|------------------------------------|--|--|--|--------|--------------|--------------|------------------------------|-----------------------|
| | | | | | | | | | | | |
| Save temp files | -t | | /SAVE_TEMP_FILES | | [options] -t | | P, M | | | | |
| Title | -title ^{2a} <i>string</i> | | | | | | P, M | | | | |
| Temporary directory | -T <i>dir-name</i> | | /TEMPORARY_FILES = <i>dir-name</i> | | [Temporary Directory] <i>directory name</i> | | P, M | | | current working directory | |
| Speed sort | -TB <i>blocksize</i> | | /TBLOCKS = <i>blocksize</i> | | [sort space] <i>blocksize</i> | | P, M | 31 | 1 | 2 | 2 |

1. Relevant only for UNIX and VMS systems that use shared memory.
2. UNIX only. 2a. UNIX X Windows only.
3. Automatically supplied with bpro and mpro commands
4. The maximum is 63KB on INTEL 8086 and 80286 processors.

5. The practical maximum is considerably lower and is system dependent.
6. Internal use only.
7. Can be used with CONNECT.
8. The practical maximum is system-dependent but is always at least 10.

- * (M) Multi-user startup commands
- * (P) Single-user startup commands
- * (S) Multi-user server/broker startup commands

| Option | UNIX/ DOS/ OS/2 | VMS | BTOS/CTOS | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default |
|---------------------|--|-----------------------------|-------------------------|--------|--------------|--------------|----------------------------|-----------------------|
| | | | | | | | | |
| Merge number | -TM <i>n</i> | /MERGE_NUM = <i>integer</i> | [merge number] <i>n</i> | P, M | 32 | 1 | 5 | 5 |
| Userid ⁷ | -U <i>userid</i> | /USER = " <i>string</i> " | [userid] <i>userid</i> | P, M | | | | |
| Video codes | -v <i>video-codes</i> (DOS & OS/2 only) | not applicable | not applicable | P, M | | | | |

1. Relevant only for UNIX and VMS systems that use shared memory.
2. UNIX only. 2a. UNIX X Windows only.
3. Automatically supplied with `bpro` and `mpro` commands
4. The maximum is 63KB on systems designed for the INTEL 8086 and 80286 processors and on BTOS/CTOS machines.

5. The practical maximum is considerably lower and is system dependent.
6. Internal use only.
7. Can be used with `CONNECT`.
8. The practical max is system dependent but is always at least 10.

- * (M) Multi-user startup commands
- * (P) Single-user startup commands
- * (S) Multi-user server/broker startup commands

STARTUP OPTIONS

The symbols you use for the option syntax are case sensitive. You must type them exactly as shown.

| Option | UNIX/ DOS/ OS/2 | VMS | BTOS/CTOS | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default |
|----------------|-----------------------|----------------------------------|----------------|--------|--------------|--------------|--|--|
| | | | | | | | | |
| ORACLE Version | -VO <i>n</i> | /ORACLE_VERSION = <i>integer</i> | not applicable | P, M | 6 | 5 | Default ORACLE version for your system | Default ORACLE version for your system |
| X Windows | -ws 2a | | | P, M | | | | |

1. Relevant only for UNIX and VMS systems that use shared memory.
2. UNIX only. 2a. UNIX X Windows only.
3. Automatically supplied with bpro and mpro commands
4. The maximum is 63KB on INTEL 8086 and 80286 processors.

5. The practical maximum is considerably lower and is system dependent.
6. Internal use only.
7. Can be used with CONNECT.
8. The practical maximum is system-dependent but is always at least 10.

- * (M) Multi-user startup commands
- * (P) Single-user startup commands
- * (S) Multi-user server/broker startup commands

| Option | UNIX/ DOS/ OS/2 | VMS | BTOS/CTOS | Usage * | Max Value | Min Value | Single- user Default | Multi-user Default |
|---------------------------|-----------------------|-------------------------------|---------------------|---------|--------------|--------------|----------------------------|-----------------------|
| | | | | | | | | |
| Extended alphabet support | -xc <i>language</i> | /COLLATE = <i>language</i> | not applicable | P, M, S | | | | |
| Statistics | -y | /STATISTICS | [options] -y | P, M | | | | |
| Century | -yy <i>n</i> | /YEAR_OFFSET = <i>integer</i> | [century] <i>n</i> | P, M | 9900 | 1100 | 1900 | 1900 |

1. Relevant only for UNIX and VMS systems that use shared memory.
2. UNIX only. 2a. UNIX X Windows only.
3. Automatically supplied with `bpro` and `mpro` commands
4. The maximum is 63KB on systems designed for the INTEL 8086 and 80286 processors and on BTOS/CTOS machines.

5. The practical maximum is considerably lower and is system dependent.
6. Internal use only.
7. Can be used with `CONNECT`.
8. The practical max is system dependent but is always at least 10.

- * (M) Multi-user startup commands
- * (P) Single-user startup commands
- * (S) Multi-user server/broker startup commands

STARTUP OPTIONS

The symbols you use for the option syntax are case sensitive. You must type them exactly as shown.

| Option | UNIX/ DOS/ OS/2 | VMS | BTOS/CTOS | Usage* | Max Value | Min Value | Single- user Default | Multi-user Default |
|--|-----------------------|----------------|------------------------|--------|--------------|--------------|----------------------------|-----------------------|
| | | | | | | | | |
| Restore 25 line mode on oper. system escape | -25 | not applicable | not applicable | P, M | | | | |
| IPC Block Size (in bytes) | not applicable | not applicable | [IPC Block Size] | S, M | 64 KB | 512 | | 2560 |
| Number of IPC Blocks | not applicable | not applicable | [Number of IPC Blocks] | S | | | | 2 per user |
| Maximum Record Size (in kilobytes) | not applicable | not applicable | [Maximum Record Size] | S | 64 | 1 | 2.5 | 2.5 |

1. Relevant only for UNIX and VMS systems that use shared memory.
2. UNIX only. 2a. UNIX X Windows only.
3. Automatically supplied with bpro and mpro commands
4. The maximum is 63KB on INTEL 8086 and 80286 processors.

5. The practical maximum is considerably lower and is system dependent.
6. Internal use only.
7. Can be used with CONNECT.
8. The practical maximum is system-dependent but is always at least 10.

- * (M) Multi-user startup commands
- * (P) Single-user startup commands
- * (S) Multi-user server/broker startup commands

ENVIRONMENT VARIABLES

| Variable Name | Description | Operating System | Default |
|---------------|--|---|--|
| DLC | Full name of the directory containing the PROGRESS software. | UNIX <input checked="" type="checkbox"/> DOS & OS/2 <input checked="" type="checkbox"/> VMS <input checked="" type="checkbox"/> BTOS <input checked="" type="checkbox"/> | /usr/dlc \DLC \$DISK1:[DLC] [sys] < dlc > |
| PATH | A list of pathnames of the directories PROGRESS searches to find DOS executable commands or programs used with the DOS statement, or to find OS/2 executable commands or programs used with OS/2, or to find UNIX executable commands or programs used with the UNIX, INPUT THROUGH, or OUTPUT THROUGH statements. | UNIX <input checked="" type="checkbox"/> DOS & OS/2 <input checked="" type="checkbox"/> VMS <input type="checkbox"/> BTOS <input type="checkbox"/> | |

(continued)

ENVIRONMENT VARIABLES

| Variable Name | Description | Operating System | Default |
|---------------|--|--|---|
| PROPATH | <p>A list of pathnames of the directories PROGRESS searches for procedures. Your current directory is indicated by a leading semicolon (DOS or OS/2) or colon (UNIX) or by two adjacent imbedded semicolons (DOS or OS/2) or colons (UNIX).</p> <p>The defaults shown for PROPATH are internally concatenated onto any value already defined externally for PROPATH.</p> | UNIX <input checked="" type="checkbox"/> | :\$DLC:\$DLC/prodemo:\$DLC/proguide |
| | | DOS & OS/2 <input checked="" type="checkbox"/> | ;%DLC%;%DLC\PRODEMO;%DLC\PROGUIDE |
| | | VMS <input checked="" type="checkbox"/> | If DLC is defined to be \$DISK1:[DLC] then “,\$DISK1:[DLC],\$DISK1:[DLC.PRODEMO], \$DISK1:[DLC.PROGUIDE]” |
| | | BTOS <input checked="" type="checkbox"/> | [sys] < dlc > :[sys] < dlc > prodemo/:[sys] < dlc > proguide/ |
| PROTERMCAP | Name of the file containing terminal definitions. | UNIX <input checked="" type="checkbox"/> | \$DLC/protermcap |
| | | DOS & OS/2 <input checked="" type="checkbox"/> | %DLC%\PROTERM.CAP |
| | | VMS <input checked="" type="checkbox"/> | DLC: PROTERM.DAT |
| | | BTOS <input checked="" type="checkbox"/> | [sys] < dlc > protermcap |

(continued)

ENVIRONMENT VARIABLES

| Variable Name | Description | Operating System | Default |
|---------------|---|--|---------|
| TERM | <p>The type of terminal you are using. Value of TERM can be returned by TERMINAL function and set during program execution with the statement</p> <p style="text-align: center;">TERMINAL=<i>termid</i>.</p> | UNIX <input checked="" type="checkbox"/> DOS & OS/2 <input type="checkbox"/> VMS <input checked="" type="checkbox"/> BTOS <input checked="" type="checkbox"/> | B20 |
| PROTERM | <p>The type of terminal you are using. Value of PROTERM can be returned by TERMINAL function and set during program execution with the statement</p> <p style="text-align: center;">TERMINAL=<i>termid</i>.</p> | UNIX <input type="checkbox"/> DOS & OS2 <input type="checkbox"/> VMS <input checked="" type="checkbox"/> BTOS <input type="checkbox"/> | none |

ENVIRONMENT VARIABLES

| Variable Name | Description | Operating System | Default |
|---------------|---|---|--|
| PROMSGS | Name of the file containing the text of the PROGRESS system messages. | UNIX <input checked="" type="checkbox"/> DOS & OS/2 <input checked="" type="checkbox"/> VMS <input checked="" type="checkbox"/> BTOS <input checked="" type="checkbox"/> | \$DLC/promsgs %DLC%\PROMSGS DLC:PROMSGS.DAT [sys] < dlc > promsgs |
| PROLOAD | The directory in which the PROBUILD product is installed. | UNIX <input checked="" type="checkbox"/> DOS & OS/2 <input checked="" type="checkbox"/> VMS <input checked="" type="checkbox"/> BTOS <input checked="" type="checkbox"/> | /dlcload \DLCLOAD \$DISK1:[DLCLOAD] [sys] < dlcload > |

ENVIRONMENT VARIABLES

| Variable Name | Description | Operating System | Default |
|---------------|--|---|---|
| PROEXE | The name of the PROGRESS executable file. This is the file that executes when you start PROGRESS. | UNIX <input checked="" type="checkbox"/> DOS & OS/2 <input checked="" type="checkbox"/> VMS <input checked="" type="checkbox"/> BTOS <input checked="" type="checkbox"/> | \$DLC/_progres %DLC%_PROGRES.EXE DLC.PROGRES.EXE [sys]< dlc > _progres |
| PROSRV | The name of the executable PROGRESS SERVER file. This is the file that executes when you start the PROGRESS database server (or broker). | UNIX <input checked="" type="checkbox"/> DOS & OS2 <input checked="" type="checkbox"/> VMS <input checked="" type="checkbox"/> BTOS <input checked="" type="checkbox"/> | \$DLC/_mprosrv %DLC%_MPROSRV.EXE DLC._MPROSRV.EXE [sys]< dlc > _mprosrv |

ENVIRONMENT VARIABLES

| Variable Name | Description | Operating System | Default |
|---------------|--|--|----------------------------|
| PROCFG | The name of the configuration file. The configuration file specifies the PROGRESS products and components you are licensed to use. | UNIX <input checked="" type="checkbox"/> | \$DLC/progress.cfg |
| | | DOS & OS/2 <input checked="" type="checkbox"/> | %DLC%\PROGRESS.CFG |
| | | VMS <input checked="" type="checkbox"/> | DLC.PROGRESS.CFG |
| | | BTOS <input checked="" type="checkbox"/> | [sys] < dlc > progress.cfg |
| PROOVL | DOS Memory Saver (DMS) only: The directory where the .OVL overlay file is stored. | UNIX <input type="checkbox"/> | %DLC%_PROGRES.OVL |
| | | DOS & OS2 <input checked="" type="checkbox"/> | |
| | | VMS <input type="checkbox"/> | |
| | | BTOS <input type="checkbox"/> | |

ENVIRONMENT VARIABLES

| Variable Name | Description | Operating System | Default |
|----------------------|---|---|--|
| DLCFT | The name of the directory into which you installed the PROGRESS FAST TRACK product. | UNIX <input checked="" type="checkbox"/> DOS & OS/2 <input checked="" type="checkbox"/> VMS <input checked="" type="checkbox"/> BTOS <input checked="" type="checkbox"/> | /usr/dlcft \DLCFT \$DISK1:[DLCFT] [sys] < dlcft > |

THE PROGRESS KEYBOARD

| Key Function | Editor | Procedure Execution | Standard Keyboard Key | Standard Control Key | | | Standard Function Key | | | Allowed in ON statement |
|----------------|--------|---------------------|-----------------------|----------------------|----------|-----------|-----------------------|----------------------------|-----------|-------------------------|
| | | | | DOS OS/2 | UNIX VMS | BTOS CTOS | DOS OS/2 | UNIX VMS | BTOS CTOS | |
| ABORT | ✓ | ✓ | | CTRL-ALT-DEL | ** | ACTION-/ | | | | |
| APPEND-LINE | ✓ | | | CTRL-A | CTRL-A | CODE-A | ALT-F2 | F12 | SHFT-F2 | |
| BACKSPACE | ✓ | ✓ | BACKSPACE | | | | | | | ✓ |
| BACK-TAB | ✓ | | SHIFT-TAB*** | CTRL-U | CTRL-U | CODE-U | | | CODE-TAB | ✓ |
| BELL | | | | | | | | | | ✓ |
| BLOCK | ✓ | | | CTRL-V | CTRL-V | CODE-V | ALT-F4 | F14 | SHIFT-F4 | |
| BOTTOM COLUMN* | | | | | | CODE-↓ | ALT-B | ESC, CURSOR DOWN ESC, B | | ✓ |

* See paragraph at end of table for explanation.

** UNIX: Ctrl-\ (Depends on UNIX stty setting for quit)
VMS: Ctrl-Y, STOP

***DOS only

(continued)

THE PROGRESS KEYBOARD

| Key Function | Editor | Procedure Execution | Standard Keyboard Key | Standard Control Key | | | Standard Function Key | | | Allowed in ON statement |
|---------------|--------|---------------------|---|----------------------|----------|-----------|-----------------------|----------|-------------|-------------------------|
| | | | | DOS OS/2 | UNIX VMS | BTOS CTOS | DOS OS/2 | UNIX VMS | BTOS CTOS | |
| BREAK-LINE | ✓ | | | CTRL-B | CTRL-B | CODE-B | ALT-F1 | F11 | SHIFT-F1 | |
| BTOS/CTOS-END | | ✓ | EXIT EXEC(BTOS) or FINISH EXEC(CTOS) PROGRESS EXIT (PROGRESS Utility to quit CONTEXT with PAUSE.) | | | | | | | |
| CANCEL PICK * | | | | | | | ALT-X | ESC, X | CODE-CANCEL | ✓ |
| CHOICES * | | | | | | CODE-C | ALT-C | ESC, C | | ✓ |
| CLEAR | ✓ | ✓ | | CTRL-Z | CTRL-Z | CODE-Z | F8 | F8 | F8 | ✓ |

* See paragraph at end of table for explanation.

(continued)

THE PROGRESS KEYBOARD

| Key Function | Editor | Procedure Execution | Standard Keyboard Key | Standard Control Key | | | Standard Function Key | | | Allowed in ON statement |
|------------------|--------|---------------------|--------------------------|----------------------|----------|--------------|-----------------------|----------|-----------|-------------------------|
| | | | | DOS OS/2 | UNIX VMS | BTOS CTOS | DOS OS/2 | UNIX VMS | BTOS CTOS | |
| CURSOR-UP | ✓ | ✓ | ↑ | CTRL-K | CTRL-K | | | | ✓ | |
| CURSOR-DOWN | ✓ | ✓ | ↓ | CTRL-J | CTRL-J | | | | ✓ | |
| CURSOR-LEFT | ✓ | ✓ | ← | | | | | | ✓ | |
| CURSOR-RIGHT | ✓ | ✓ | → | CTRL-L | CTRL-L | | | | ✓ | |
| DELETE-CHARACTER | ✓ | ✓ | DEL DELETE(BTOS/CTOS) | | | | | | ✓ | |
| DELETE-COLUMN* | | | | | | CODE-SHIFT-D | ALT-Z | ESC,Z | ✓ | |

* See paragraph at end of table for explanation.

(continued)

THE PROGRESS KEYBOARD

| Key Function | Editor | Procedure Execution | Standard Keyboard Key | Standard Control Key | | | Standard Function Key | | | Allowed in ON statement |
|--------------|--------|---------------------|---|----------------------|----------|-----------|-----------------------|----------|-----------|-------------------------|
| | | | | DOS OS/2 | UNIX VMS | BTOS CTOS | DOS OS/2 | UNIX VMS | BTOS CTOS | |
| DELETE-FIELD | | | | | | CODE-J | ALT-D | ESC,D | | ✓ |
| DELETE-LINE | ✓ | | | CTRL-D | CTRL-D | CODE-D | F10 | F10 | F10 | |
| DOS-END | | ✓ | | Type EXIT | | | | | | |
| ENDKEY | | ✓ | | | | | | | | ✓ |
| END-ERROR | ✓ | ✓ | ESC (DOS & OS/2) ₁ CANCEL or FINISH | CTRL-E | CTRL-E | CODE-E | F4 | F4 | F4 | ✓ |
| ERROR | | ✓ | | | | | | | | ✓ |
| FIND | ✓ | | | CTRL-F | CTRL-F | CODE-F | ALT-F3 | F13 | SHIFT-F3 | |

* See paragraph at end of table for explanation.

1. BTOS/CTOS

(continued)

THE PROGRESS KEYBOARD

| Key Function | Editor | Procedure Execution | Standard Keyboard Key | Standard Control Key | | | Standard Function Key | | | Allowed in ON statement |
|-----------------|--------|---------------------|-----------------------|----------------------|----------|--------------|-----------------------|----------|----------------|-------------------------|
| | | | | DOS OS/2 | UNIX VMS | BTOS CTOS | DOS OS/2 | UNIX VMS | BTOS CTOS | |
| GET | ✓ | | | CTRL-G | CTRL-G | CODE-G | F5 | F5 | F5 | |
| GO | ✓ | ✓ | | CTRL-X | CTRL-X | CODE-X | F1 | F1 | F1 | ✓ |
| GOTO * | | | | | | CODE-SHIFT-G | ALT-G | ESC,G | | ✓ |
| HELP | ✓ | ✓ *** | HELP | CTRL-W | CTRL-W | CODE-W | F2 | F2 | F2 | ✓ |
| HOME | ✓ | ✓ | HOME | | ESC-H | | | | CODE-NEXT-PAGE | ✓ |
| INSERT COLUMN * | | | | | | CODE-SHIFT-C | ALT-N | ESC,N | | ✓ |
| INSERT FIELD * | | | | | | CODE-I | ALT-I | ESC,I | | ✓ |

* See paragraph at end of table for explanation.

*** Allowed only if help procedure, applhelp.p., exists.

(continued)

THE PROGRESS KEYBOARD

| Key Function | Editor | Procedure Execution | Standard Keyboard Key | Standard Control Key | | | Standard Function Key | | | Allowed in ON statement |
|----------------------|--------|---------------------|---|----------------------|----------|--------------|-----------------------|----------|-----------|-------------------------|
| | | | | DOS OS/2 | UNIX VMS | BTOS CTOS | DOS OS/2 | UNIX VMS | BTOS CTOS | |
| INSERT-FIELD DATA* | | | | | | CODE-SHIFT-L | ALT-F | ESC,F | | ✓ |
| INSERT-FIELD * LABEL | | | | | | | ALT-E | ESC,E | | ✓ |
| INSERT-MODE | ✓ | ✓ | INSERT | CTRL-T | CTRL-T | CODE-T | F3 | F3 | F3 | ✓ |
| LEFT-END | ✓ | | CTRL-← (DOS & OS/2) ESC - ← (UNIX & VMS) | | | | | | CODE-← | ✓ |
| MAIN MENU * | | | | | | CODE-M | ALT-M | ESC,M | | ✓ |
| MOVE * | | | | | | MOVE | ALT-V | ESC,V | | ✓ |
| NEW-LINE | ✓ | | | CTRL-N | CTRL-N | CODE-N | F9 | F9 | F9 | |
| PAGE-DOWN | ✓ | | NEXT PAGE or PG DN | | | | ALT-F6 | F16 | SHIFT-F6 | |

* See paragraph at end of table for explanation.

THE PROGRESS KEYBOARD

| Key Function | Editor | Procedure Execution | Standard Keyboard Key | Standard Control Key | | | Standard Function Key | | | Allowed in ON statement |
|------------------|--------|---------------------|-----------------------|----------------------|----------|--------------|-----------------------|----------|-----------|-------------------------|
| | | | | DOS OS/2 | UNIX VMS | BTOS CTOS | DOS OS/2 | UNIX VMS | BTOS CTOS | |
| PAGE-UP | ✓ | | PREV PAGE or PG UP | | | | ALT-F5 | F15 | SHIFT-F5 | |
| PICK* | | | | | | CODE-K | ALT-P | ESC,P | | ✓ |
| PICK-AREA* | | | | | | CODE-SHIFT-A | ALT-W | ESC,W | | ✓ |
| PICK-LABEL-DATA* | | | | | | CODE-Q | ALT-Q | ESC,Q | | ✓ |
| PUT | ✓ | | | CTRL-P | CTRL-P | CODE-P | F6 | F6 | F6 | |
| RECALL | ✓ | ✓ | | CTRL-R | CTRL-R | CODE-R | F7 | F7 | F7 | ✓ |
| REPAINT | ✓ | | | ALT-P | ESC,P | | | | | ✓ |
| REPORTS* | | | | | | CODE-SHIFT-T | ALT-A | ESC,A | | ✓ |

* See paragraph at end of table for explanation.

(continued)

THE PROGRESS KEYBOARD

| Key Function | Editor | Procedure Execution | Standard Keyboard Key | Standard Control Key | | | Standard Function Key | | | Allowed in ON statement |
|----------------|--------|---------------------|---|----------------------|------------------------|---------------|-----------------------|----------|-----------|-------------------------|
| | | | | DOS OS/2 | UNIX VMS | BTOS CTOS | DOS OS/2 | UNIX VMS | BTOS CTOS | |
| RESUME-DISPLAY | ✓ | ✓ | | CTRL-Q | CTRL-Q | | | | | |
| RETURN | ✓ | ✓ | RETURN | CTRL-M | CTRL-M | | | | | ✓ |
| RIGHT-END | ✓ | | CTRL-→ (DOS & OS/2) ESC - → (UNIX & VMS) | CTRL-E | CTRL-E | | F4 | F4 | CODE-→ | ✓ |
| SEARCH | ✓ | | | | | CODE-F | ALT-F | ESC,F | | |
| SCROLL-LEFT * | | | | | | | ALT-L | ESC,L | SHIFT-← | ✓ |
| SCROLL-RIGHT * | | | | | | | ALT-R | ESC,R | SHIFT-→ | ✓ |
| SETTINGS * | | | | | | CODE-S | ALT-S | ESC,S | | ✓ |
| STOP | | ✓ | | BREAK | CTRL-C ^{****} | ACTION-CANCEL | | | | ✓ |

* See paragraph at end of table for explanation.

**** Depends on UNIX stty setting for intr

THE PROGRESS KEYBOARD

| Key Function | Editor | Procedure Execution | Standard Keyboard Key | Standard Control Key | | | Standard Function Key | | | Allowed in ON statement |
|--------------|--------|---------------------|-----------------------|----------------------|--------------------------------|-----------|-----------------------|----------|-----------|-------------------------|
| | | | | DOS OS/2 | UNIX VMS | BTOS CTOS | DOS OS/2 | UNIX VMS | BTOS CTOS | |
| | | | | DOS OS/2 | UNIX VMS | BTOS CTOS | DOS OS/2 | UNIX VMS | BTOS CTOS | |
| STOP-DISPLAY | ✓ | ✓ | | CTRL-S | CTRL-S | | | | | |
| TAB | ✓ | ✓ | TAB | CTRL-I | CTRL-I | | | | | ✓ |
| TOP-COLUMN * | | | | ALT-T | ESC, CURSOR UP ESC, T | | | | | ✓ |

* See paragraph at end of table for explanation.

THE PROGRESS KEYBOARD

| Key Function | Editor | Procedure Execution | Standard Keyboard Key | Standard Control Key | | | Standard Function Key | | | Allowed in ON statement |
|--------------|--------|---------------------|-----------------------|----------------------|---------------|-----------|-----------------------|----------|-----------|-------------------------|
| | | | | DOS OS/2 | UNIX VMS | BTOS CTOS | DOS OS/2 | UNIX VMS | BTOS CTOS | |
| UNIX-END | | ✓ | | | * * CTRL-D | | | | | |
| VMS-END | | ✓ | | | LOGOUT | | | | | |

* * Depends on UNIX stty setting for eof

* These key functions do not have automatic actions associated with them when you use them in the editor or while running a procedure. However, these key functions are available for use with the ON statement together with the KEYFUNCTION and LASTKEY functions. For example, one of the special key functions marked with an asterisk in the tables above is CHOICES. The following statement check to see if the user pressed the CHOICES key. The example defines the F2 key as the CHOICES key. If the user presses F2, the procedure might then display a list of available choices.

KEYCODES AND KEY LABELS

| Key Code | Key Label | Allowed in ON Statement or GO-ON Phrase | | |
|----------|--------------------------|--|-------|-----|
| | | DOS & OS/2 | UNIX | VMS |
| 0 | CTRL-@ | ✓ | ✓ (1) | ✓ |
| 1 - 7 | CTRL-A through CTRL-G | ✓ | ✓ (1) | ✓ |
| 8 | BACKSPACE | ✓ | ✓ (1) | ✓ |
| 9 | TAB | ✓ | ✓ (1) | ✓ |
| 10 - 12 | CTRL-J through CTRL-L | ✓ | ✓ (1) | ✓ |

(1) - Unless pre-empted by UNIX stty settings for intr, quit, stop display, resume display.

| Key Code | Key Label | Allowed in ON Statement or GO-ON Phrase | | |
|----------|----------------------------------|--|----------------|--------|
| | | DOS & OS/2 | UNIX | VMS |
| 13 | ENTER (DOS/OS2) RETURN (UNIX) | ✓ ✓ | ✓ (1) ✓ (1) | ✓ ✓ |
| 14 - 26 | CTRL-N through CTRL-Z | ✓ | ✓ (1) | ✓ |
| 27 | ESC | ✓ | ✓ (1) | ✓ |
| 28 | CTRL-\ | ✓ | ✓ (1) | ✓ |
| 29 | CTRL-] | ✓ | ✓ (1) | ✓ |

(continued)

BTOS/CTOS keycodes can be found in
Chapter 2 of the *PROGRAMMING Handbook*.

KEYCODES AND KEY LABELS

| Key Code | Key Label | Allowed in ON Statement or GO-ON Phrase | | |
|-----------|---|--|-------|-----|
| | | DOS & OS/2 | UNIX | VMS |
| 30 | CTRL-^ | ✓ | ✓ (1) | ✓ |
| 31 | CTRL- _ | ✓ | ✓ (1) | ✓ |
| 32 - 126 | Corresponding extended ASCII character | | ✓ (1) | ✓ |
| 127 | DEL | ✓ | ✓ (1) | ✓ |
| 128 - 255 | Corresponding extended ASCII character | | | |

- (1) - Unless pre-empted by UNIX stty settings for intr, quit, stop display, resume display.
 (2) - Only if key is defined in protermcap file.
 (3) - Corresponds to Alt-F1 through Alt-F10 on the DOS keyboard.

| Key Code | Key Label | Allowed in ON Statement or GO-ON Phrase | | |
|-----------|-------------|--|-------|-------|
| | | DOS & OS/2 | UNIX | VMS |
| 256 - 299 | null string | | | |
| 301 - 310 | F1 - F10 | ✓ | ✓ (2) | ✓ (6) |
| 311 - 320 | F11 - F20 | ✓ (3) | ✓ (2) | ✓ |
| 321 - 330 | F21 - F30 | ✓ (4) | ✓ (2) | ✓ |
| 331 - 340 | F31 - F40 | ✓ (5) | ✓ (2) | ✓ |

(continued)

- (4) - Corresponds to Shift-F1 through Shift F-10 on the DOS keyboard.
 (5) - Corresponds to Ctrl-F1 through Ctrl-F10 on the DOS keyboard.
 (6) - Not allowed for vt100 terminals.

KEYCODES AND KEY LABELS

| Key Code | Key Label | Allowed in ON Statement or GO-ON Phrase | | |
|-----------|--------------|--|-------|-----|
| | | DOS & OS/2 | UNIX | VMS |
| 341 - 399 | F41 - F99 | | ✓ (2) | ✓ |
| 400 - 499 | PF0 - PF99 | | ✓ (2) | ✓ |
| 501 | CURSOR-UP | ✓ | ✓ (2) | ✓ |
| 502 | CURSOR-DOWN | ✓ | ✓ (2) | ✓ |
| 503 | CURSOR-RIGHT | ✓ | ✓ (2) | ✓ |

(2) - Only if key is defined in protermcap file.

| Key Code | Key Label | Allowed in ON Statement or GO-ON Phrase | | |
|----------|-------------|--|-------|-----|
| | | DOS & OS/2 | UNIX | VMS |
| 504 | CURSOR-LEFT | ✓ | ✓ (2) | ✓ |
| 505 | HOME | ✓ | ✓ (2) | ✓ |
| 506 | END | ✓ | ✓ (2) | ✓ |
| 507 | PAGE-UP | ✓ | ✓ (2) | ✓ |
| 508 | PAGE-DOWN | ✓ | ✓ (2) | ✓ |
| 509 | BACK-TAB | ✓ | ✓ (2) | ✓ |

(continued)

BTOS/CTOS keycodes can be found in
Chapter 2 of the *PROGRAMMING Handbook*.

KEYCODES AND KEY LABELS

| Key Code | Key Label | Allowed in ON Statement or GO-ON Phrase | | |
|----------|-----------|--|-------|-----|
| | | DOS & OS/2 | UNIX | VMS |
| 510 | INS | | ✓ (2) | ✓ |
| 511 | HELP | | ✓ (2) | ✓ |
| 512 | DEL-CHAR | | ✓ (2) | ✓ |
| 513 | EXECUTE | | ✓ (2) | ✓ |
| 514 | PAGE | | ✓ (2) | ✓ |
| 515 | FIND | | ✓ (2) | ✓ |
| 516 | INS-LINE | | ✓ (2) | ✓ |
| 517 | DEL-LINE | | ✓ (2) | ✓ |

(2) - Only if key is defined in protermcap file.

| Key Code | Key Label | Allowed in ON Statement or GO-ON Phrase | | |
|----------|--------------|--|-------|-----|
| | | DOS & OS/2 | UNIX | VMS |
| 518 | LINE-ERASE | | ✓ (2) | ✓ |
| 519 | PAGE-ERASE | | ✓ (2) | ✓ |
| 520 | CTRL-BREAK | | ✓ (2) | ✓ |
| 521 | CTRL-ALT-DEL | | ✓ (2) | ✓ |
| 522 | EXIT | | ✓ (2) | ✓ |
| 523 | CTRL-RIGHT | ✓ | ✓ (2) | ✓ |
| 524 | CTRL-LEFT | ✓ | ✓ (2) | ✓ |

BTOS/CTOS keycodes can be found in
Chapter 2 of the *PROGRAMMING Handbook*.

(continued)

KEYCODES AND KEY LABELS

| Key Code | Key Label | Allowed in ON Statement or GO-ON Phrase | | |
|----------|-----------|--|-------|-----|
| | | DOS & OS/2 | UNIX | VMS |
| 525 | U1 | | ✓ (2) | ✓ |
| 526 | U2 | | ✓ (2) | ✓ |
| 527 | U3 | | ✓ (2) | ✓ |
| 528 | U4 | | ✓ (2) | ✓ |
| 529 | U5 | | ✓ (2) | ✓ |
| 530 | U6 | | ✓ (2) | ✓ |
| 531 | U7 | | ✓ (2) | ✓ |
| 532 | U8 | | ✓ (2) | ✓ |

(2) - Only if key is defined in protermcap file.

| Key Code | Key Label | Allowed in ON Statement or GO-ON Phrase | | |
|----------|-----------|--|-------|-----|
| | | DOS & OS/2 | UNIX | VMS |
| 533 | U9 | | ✓ (2) | ✓ |
| 534 | U10 | | ✓ (2) | ✓ |
| 535 | ERASE | | ✓ (2) | ✓ |
| 536 | WHITE | | ✓ (2) | ✓ |
| 537 | BLUE | | ✓ (2) | ✓ |
| 538 | RED | | ✓ (2) | ✓ |
| 539 | RESET | | ✓ (2) | ✓ |

BTOS/CTOS keycodes can be found in
Chapter 2 of the *PROGRAMMING Handbook*.

(continued)

KEYCODES AND KEY LABELS

| Key Code | Key Label | Allowed in ON Statement or GO-ON Phrase | | |
|----------|-----------|--|-------|-----|
| | | DOS & OS/2 | UNIX | VMS |
| 540 | ESC-F | | ✓ (2) | ✓ |
| 541 | ESC-N | | ✓ (2) | ✓ |
| 542 | ESC-1 | | ✓ (2) | ✓ |
| 543 | ESC-2 | | ✓ (2) | ✓ |
| 544 | ESC-3 | | ✓ (2) | ✓ |
| 545 | ESC-4 | | ✓ (2) | ✓ |
| 546 | ESC-5 | | ✓ (2) | ✓ |
| 547 | ESC-6 | | ✓ (2) | ✓ |

| Key Code | Key Label | Allowed in ON Statement or GO-ON Phrase | | |
|----------|----------------|--|-------|-----|
| | | DOS & OS/2 | UNIX | VMS |
| 548 | ESC-7 | | ✓ (2) | ✓ |
| 549 | ESC-8 | | ✓ (2) | ✓ |
| 550 | ESC-9 | | ✓ (2) | ✓ |
| 551 | ESC-Z | | ✓ (2) | ✓ |
| 552 | ESC-LEFT-ARROW | | ✓ (2) | ✓ |
| 553 | ESC-UP-ARROW | | ✓ (2) | ✓ |
| 554 | ESC-DOWN-ARROW | | ✓ (2) | ✓ |
| 555 | ESC-V | | ✓ (2) | ✓ |

(2) - Only if key is defined in protermcap file.

ALTERNATE KEY LABELS

| Key Code | Alternate Key Labels |
|----------|--------------------------------|
| 7 | BELL |
| 8 | BS |
| 10 | LINEFEED, LF |
| 12 | FORMFEED, FF |
| 13 | RETURN (DOS), ENTER (UNIX), CR |
| 27 | ESCAPE |
| 127 | CANCEL |
| 501 | UP |
| 502 | DOWN |

(continued)

ALTERNATE KEY LABELS

| Key Code | Alternate Key Labels |
|-----------------|--------------------------------------|
| 503 | RIGHT |
| 504 | LEFT |
| 505 | ESC-H |
| 507 | PGUP, PREV-PAGE, PREV-SCRN |
| 508 | PGDN, NEXT-PAGE, NEXT-SCRN |
| 509 | SHIFT-TAB |
| 510 | INSERT, INS-CHAR, INS-C, INSERT-HERE |
| 512 | DELETE, DELETE-CHAR, DEL-C |
| 516 | INS-L, LINE-INS |
| 517 | DEL-L, LINE-DEL |

THE PROGRESS LANGUAGE

This section of *Pocket PROGRESS* contains an alphabetical list of PROGRESS statements, phrases, functions and operators. Each item is followed by a description of that item and the appropriate PROGRESS syntax.

EXAMPLE:

| | | | | | |
|--|---|---|-----------------------------|---|---------------|
| Statement, Phrase Function, or Operator | → | CREATE <i>Statement</i> | CREATE <i>record</i> | ← | Syntax |
| Description | → | Creates a record in a file, setting all the fields in the record to their default initial values and moving a copy of that record to a record buffer. | | | |

PROGRESS SYNTAX RULES

- Uppercase words are required keywords. Although they are always shown in uppercase, you can use either uppercase or lowercase when using them in a procedure.
- Italics identify parameters, or arguments, that you supply.
- End all statements (except for DO, FOR EACH, and REPEAT) with a period. End DO, FOR EACH, and REPEAT statements with either a period or a colon.
- Parentheses surrounding parameters or arguments indicate that the parentheses () are required as part of the syntax.
- Square brackets indicate that an item is optional, with the exception of array references, where you actually type the square brackets [].
- Braces indicate that an item is required, with the exception of include procedures and argument references, where you actually type the braces { }.

- Ellipses (...) indicate that you can choose one or more of the items they follow. If a group of items is enclosed in brackets and is followed by ellipses, you can optionally choose one or more of those items. If a group of items is enclosed in braces and is followed by ellipses, you must choose one or more of those items. A comma followed by ellipses (, ...) indicates that you must place commas between items in a list.
- When you see *aggregate-phrase*, *color-phrase*, *editing-phrase*, *format-phrase*, *frame-phrase* or *record-phrase* in a syntax diagram, refer to the appropriate page in this section for a description of that phrase.
- Expressions are used in many PROGRESS statements and functions. An expression is a constant, field name, variable name, or any combination of these.
- A string is a character constant, enclosed in quotes.

PROGRESS SYNTAX SUMMARY

- : *Punctuation*
Ends block labels and block header statements
(DO, FOR EACH, REPEAT).

- ; *Special Character*
When combined with a second character in the
PROGRESS procedure editor, provides alternative
representations of special PROGRESS characters.

- ; *Punctuation*
In PROGRESS Version 6 and later versions, can be used
to terminate statements when -Q is turned on for the session
in which the procedure is compiled. Just like a period.
This disables the use of the semicolon within, for example,
Unix escapes such as "UNIX SMBL = foo: export SMBL".

- . *Punctuation*
Ends all statements including block header statements and block labels.

- , *Punctuation*
Separates multiple file specifications (used in FOR EACH statements and PRESELECT phrases), multiple branching phrases (used in UNDO statements and phrases), and multiple argument of a function.

- ? *Special Character*
Represents the unknown value.

- \ *Special Character*
An “escape” character (UNIX only). – see also ~ (tilde).
A directory path separator (DOS and OS/2 only).

~ *Special Character*

An "escape" character that tells PROGRESS to read the following character literally and not to give that character special meaning. A tilde followed by three octal digits represents a single character.

" *Special Character*

Encloses character constants or strings. To use quotes within a quoted character string, you must either use two double quotes (" "), which compile to a single double quote ("), or you must put a tilde (~) in front of any quotes within the quoted character string.

' *Special Character*

The single quote functions exactly as the double quote. However, if you use both single and double quotes in a statement, the compiler checks the outermost quotes first, giving them precedence over the innermost quotes.

/ *Special Character*

A directory path separator (UNIX). Also used for date fields.

() *Expression Precedence*

Raises expression precedence. Also, some functions require you to enclose arguments in parentheses.

[] *Array Reference*

Encloses array subscripts (such as [1], [2], etc.) or ranges (such as [1 FOR 4]).

{ n }
{ & argument-name }

{ } *Argument Reference*

Refers to an argument being passed by a calling procedure, or to an argument in an include file.
{*} refers to all arguments being passed.

YES, NO, TRUE, FALSE

Logical Value

Represent values of logical fields or variables.

/* This is a Comment */

Puts explanatory text into a procedure.

PROGRESS ignores text between the characters /* and */.

+ **Unary Positive Operator**

Preserves the positive or negative value of a numeric expression.

+ expression

+ **Addition Operator**

Adds two numeric expressions.

expression + expression

+ **Concatenation Operator**

Produces a character value by joining, or concatenating, two character strings or expressions.

expression + expression

+ Date Addition Operator

Adds a number of days to a date, producing a date result.

date + days

- Unary Negative Operator

Reverses the sign of a numeric expression.

- expression

- Subtraction Operator

Subtracts one numeric expression from another numeric expression.

expression - expression

- Date Subtraction Operator

Subtracts a number of days from a date, producing a date result, or subtracts one date from another, producing an integer result representing the number of days between two dates.

$$date - \left\{ \begin{array}{l} days \\ date \end{array} \right\}$$

*** Multiplication Operator**

Multiplies two numeric expressions.

$$expression * expression$$

/ Division Operator

Divides one numeric expression by another numeric expression, producing a decimal result.

$$expression / expression$$

= or **EQ Operator**

Returns a TRUE value if two expressions are equal.

expression { $\begin{matrix} \text{EQ} \\ = \end{matrix}$ } *expression*

= **Assignment Statement**

Assigns the value of an expression to a database field or variable.

field = *expression*

< or **LT Operator**

Returns a TRUE value if the first of two expressions is less than the second expression.

expression { $\begin{matrix} \text{LT} \\ < \end{matrix}$ } *expression*

< = or **LE Operator**

Returns a TRUE value if the first of two expressions is less than or equal to the second expression.

$$expression \left\{ \begin{array}{c} \text{LE} \\ < = \end{array} \right\} expression$$
> or **GT Operator**

Returns a TRUE value if the first of two expressions is greater than the second expression.

$$expression \left\{ \begin{array}{c} \text{GT} \\ > \end{array} \right\} expression$$
> = or **GE Operator**

Returns a TRUE value if the first of two expressions is greater than or equal to the second expression.

$$expression \left\{ \begin{array}{c} \text{GE} \\ = \end{array} \right\} expression$$

< > or NE Operator

Compares two expressions and returns a TRUE value if they are not equal.

expression { ^{NE}
_{< >} } *expression*

ACCUM Function

Returns the value of an aggregate expression that has been calculated by an ACCUMULATE or DISPLAY statement.

ACCUM *aggregate–phrase expression*

ACCUMULATE Statement

Calculates one or more aggregate values of an expression during the iterations of a block. Use the ACCUM function to access the result of this accumulation.

ACCUMULATE { *expression (aggregate–phrase)* } ...

Aggregate *Phrase*

Identifies one or more values to be calculated based on a change in an expression or break group.

| | | | | | |
|---|--|---|-----|---------------------------|-----|
| { | AVERAGE COUNT MAXIMUM MINIMUM TOTAL SUB-AVERAGE SUB-COUNT SUB-MAXIMUM SUB-MINIMUM SUB-TOTAL | } | ... | [BY <i>break-group</i>] | ... |
|---|--|---|-----|---------------------------|-----|

ALIAS *Function*

The ALIAS function returns the alias corresponding to the integer value of expression.

ALIAS (*integer-expression*)

ALTER TABLE (SQL) Statement

Adds new columns to a table, deletes columns from a table, or changes the format or labels associated with an existing column.

ALTER TABLE *table-name* { **ADD COLUMN** *column-name datatype*
[**FORMAT** *string*] [**LABEL** *string*]
[**COLUMN-LABEL** *string* [! *string*] ...]
[[**NOT**] **CASE-SENSITIVE**]
[**DEFAULT** *initial-value*]
DROP COLUMN *column-name*
ALTER COLUMN *column-name*
[**FORMAT** *string*] [**LABEL** *string*]
[**COLUMN-LABEL** *string* [! *string*] ...]
[[**NOT**] **CASE-SENSITIVE**]
[**DEFAULT** *initial-value*] }

AMBIGUOUS *Function*

Returns a TRUE value if the last FIND statement for a particular record found more than one record that met the index criteria specified.

AMBIGUOUS *record*

AND *Operator*

Returns a TRUE value if each of two logical expressions is TRUE.

expression AND expression

APPLY *Statement*

In an EDITING Phrase, performs the function of a specified integer keyboard key code. Outside an EDITING Phrase, the expression you APPLY can represent one of HELP, END-ERROR, ERROR, ENDKEY, or STOP.

APPLY *expression*

ASC Function

Converts a character expression representing a single character into its corresponding ASCII integer value.

ASC (expression)

ASSIGN Statement

Moves data previously placed in a screen buffer, usually by a PROMPT-FOR statement, to the corresponding fields and variables.

ASSIGN { *field*
field = *expression* } ...

ASSIGN record [EXCEPT *field* ...]

AVAILABLE Function

Returns a TRUE value if the named record buffer contains a record and returns a FALSE value if the record buffer is empty.

AVAILABLE record

BEGINS *Function*

Tests a character expression to see if that expression begins with a second character expression.

expression1 **BEGINS** *expression2*

BELL *Statement*

Causes the terminal to “beep” if the terminal is the current output destination.

BELL

BTOS Statement

Runs a program, BTOS command, BTOS submit file, or start the BTOS executive to allow interactive processing of BTOS commands.

BTOS [SILENT]

btos-command

OS-APPEND *file-expression-from* *file-expression-to*

OS-COPY *file-expression-from* *file-expression-to*

OS-DELETE *filename-expression...*

OS-RENAME *oldname-expression* *newname-expression*

OS-REQUEST *Cd Erc nC nRq nRs C1..Cn Rq1..Rqn Rs1..Rsn*

[run-file *[command* *[argument* *]* *]*

<run-file *[* *[VALUE(expression)* *]* *...* *]*

SUBMIT *submit-file-spec* *[parameter-list ...* *]*

CALL *Statement*

Transfers control to a dispatch program which calls a C routine that you have written using PROGRESS HLC.

CALL *routine-identifier* [*argument*] ...

CAN-DO *Function*

Compares the current userid with a list of users that have permission to access a specified file. Returns a TRUE value if the userid matches an entry in the list. You generally use the CAN-DO function to do security checking.

CAN-DO (*idlist* [, *string*])

CAN-FIND Function

Returns a TRUE value if a record can be found which meets the specified FIND criteria. CAN-FIND does not actually make the record available to the procedure.

CAN-FIND $\left[\begin{array}{l} \text{FIRST} \\ \text{LAST} \end{array} \right]$ *record* [*constant*] $\left[\begin{array}{l} \text{WHERE } \textit{expression} \\ \text{USING } \textit{field} \text{ [AND } \textit{field} \text{] } \dots \\ \text{OF } \textit{record} \\ \text{USE-INDEX } \textit{index} \end{array} \right] \dots)$

CAPS Function

Converts any lowercase letters in a character string expression to uppercase letters, and returns the resulting character string.

CAPS(*expression*)

CHOOSE *Statement*

Moves a highlight bar among a series of choices and selects a choice when the user either presses RETURN or enters a unique combination of characters.

$$\text{CHOOSE } \left\{ \begin{array}{l} \text{ROW } \textit{field} \\ \text{FIELD } \left\{ \textit{field} \dots [\text{HELP } \textit{char-constant}] \right\} \dots \end{array} \right\}$$

$$\left[\begin{array}{l} \text{AUTO-RETURN} \\ \text{COLOR } \textit{color-phrase} \\ \text{GO-ON } (\textit{key-label}) \dots \\ \text{KEYS } \textit{char-variable} \\ \text{NO-ERROR} \\ \text{PAUSE } \textit{expression} \end{array} \right] \dots [\textit{frame-phrase}]$$
CHR *Function*

Converts an ASCII integer value to its corresponding character value.

$$\text{CHR}(\textit{expression})$$

CLEAR Statement

Clears the data and colors (and side-labels for a down frame) displayed in a frame.

CLEAR [FRAME *frame*] [ALL] [NO-PAUSE]

CLOSE (SQL) Statement

Closes an open cursor.

CLOSE *cursor-name*

COLOR Phrase

Specifies a video attribute or color.

The *fgnd-color* and *bgnd-color* specifications apply only on DOS.

The *protermcap-attribute* applies only on UNIX and VMS.

| | | |
|---|--|---|
| { | NORMAL INPUT MESSAGES <i>protermcap-attribute</i> <i>dos-hex-attribute</i> [BLINK-] [BRIGHT] [<i>fgnd-color</i>] [/ <i>bgnd-color</i>] [BLINK-] [RVV-] [UNDERLINE-] [BRIGHT-] [<i>fgnd-color</i>] VALUE (<i>expression</i>) | } |
|---|--|---|

COLOR Statement

Indicates the video attribute or color to use for normal display, or to use when a field is ready for data entry.

COLOR { [DISPLAY] *color-phrase* } ... *field* ... [*frame-phrase*]
 PROMPT *color-phrase* }

COMMIT OFF (SQL) Statement

Disables the SQL statement COMMIT WORK and enables the PROGRESS transaction management facilities.

COMMIT OFF

COMMIT ON (SQL) Statement

Enables the SQL Statement COMMIT WORK and disables the PROGRESS transaction management facilities.

COMMIT ON

COMMIT STATUS (SQL) Statement

Displays a message stating whether the SQL commit/rollback is enabled or disabled.

COMMIT STATUS

COMMIT WORK (*SQL*) *Statement*

Commits all database changes effected by SQL data manipulation statements since the previous **COMMIT WORK** or **ROLLBACK** work statement on since the beginning of the session.

COMMIT WORK

COMPILE Statement

Compiles a procedure. A compilation lasts for a session (a “session” compile) or can be saved permanently for use in a later session (in an “object” or “.r” file).

COMPILE { *procedure* } [ATTR-SPACE
 VALUE (*expression*)] [NO-ATTR-SPACE]

[SAVE [INTO { *directory* | value(*expression*) }]
 LISTING { *listfile* } [APPEND
 VALUE (*expression*)] [PAGE-SIZE *n*
 PAGE-WIDTH *n*] ...
 XREF *filename* [APPEND]
 XCODE *expression*] ...

CONNECT Statement

Allows access to one or more databases from within a PROGRESS procedure.

CONNECT { [*physical-name*]
 [*options*] } [NO-ERROR]

CONNECTED *Function*

Lets you determine whether a database is connected. If *logical name* is the logical name or *alias* is the alias of a connected database, then the CONNECTED function returns TRUE; otherwise, it returns FALSE.

CONNECTED({ *logical -name* }
 { *alias* })

COUNT-OF *Function*

Returns an integer that is the total number of records in the file or files you are using for a break group.

COUNT-OF (break-group)

CREATE *Statement*

Creates a record in a file, sets all the fields in the record to their default initial values and moves a copy of that record to a record buffer.

CREATE record

CREATE ALIAS *Statement*

Creates an alias for a database. The alias is added to a table of existing aliases.

```
CREATE ALIAS alias FOR DATABASE logical name [NO-ERROR].
```

CREATE INDEX (*SQL*) *Statement*

Creates an index.

```
CREATE [UNIQUE] INDEX index-name ON table-name (column-list)
```

CREATE TABLE (*SQL*) *Statement*

Creates a new base table containing the columns you specify.

```
CREATE TABLE table-name {
  ( { column-name datatype
    [ NOT NULL [ UNIQUE ] ]
    [ FORMAT string ] [ LABEL string ]
    [ COLUMN-LABEL string [! string] ...]
    [ [ NOT] CASE-SENSITIVE ]
    [ DEFAULT initial value ] }
  | { UNIQUE ( column-name [,...] ) } } [,...]
```

CREATE VIEW (*SQL*) *Statement*

Creates a viewed table (view) from one or more base tables and/or other views.

```
CREATE VIEW view-name [ (column-list) ] AS SELECT-statement
[WITH CHECK OPTION ]
```


CTOS Statement

Runs a program, CTOS command, CTOS submit file, or starts the CTOS executive to allow interactive processing of CTOS commands.

CTOS [SILENT]

```
ctos-command  
OS-APPEND file-expression-from file-expression-to  
OS-COPY file-expression-from file-expression-to  
OS-DELETE filename-expression...  
OS-RENAME oldname-expression newname-expression  
OS-REQUEST Cd Erc nC nRq nRs C1..Cn Rq1..Rqn Rs1..Rsn  
  
[run-file [command [argument ] ]  
<run-file [ [VALUE(expression) ] ... ]  
SUBMIT submit-file-spec [parameter-list ... ]
```

CREATE VIEW (*SQL*) *Statement*

Creates a viewed table (view) from one or more base tables and/or other views.

CREATE VIEW *view-name* [(*column-list*)] **AS** *SELECT-statement*

DATE *Function*

Converts three integer values representing a month, day, and year, into a date. The year includes the century.

DATE (*month, day, year*)

DAY *Function*

Converts a date to a day of the month integer value from 1 to 31.

DAY (*date*)

DBNAME *Function*

Returns the physical name of your first connected database.

DBNAME

DBRESTICTIONS *Function*

Returns a character string that describes features that are not supported for this database.

DBRESTRICTIONS { *(integer expression)*
(logical-name)
(alias) }

DBTYPE *Function*

Returns the database type of a currently connected database (“PROGRESS”, “RMS”, “ORACLE”, etc.) DBTYPE accepts as a parameter either an integer expression or a character expression.

DBTYPE { *(integer expression)*
(logical-name)
(alias) }

DBVERSION *Function*

Returns a “5” if a connected database is a Version 5 database and a “6” if it is a Version 6 database. For non-PROGRESS databases, you see the appropriate version number of your database. DBVERSION accepts as a parameter either an integer expression or a character expression.

$$\text{DBVERSION} \left\{ \begin{array}{l} \text{(integer expression)} \\ \text{(logical-name)} \\ \text{(alias)} \end{array} \right\}$$
DECIMAL *Function*

Converts an expression of any data type to a decimal value.

DECIMAL (*expression*)

DECLARE CURSOR (*SQL*) *Statement*

Associates a cursor name with a SELECT statement.

DECLARE *cursor-name* CURSOR FOR *SELECT-statement*

DEFINE BUFFER *Statement*

PROGRESS automatically uses one record buffer per file to store one record at a time from that file as needed in a given procedure. The DEFINE BUFFER statement defines additional buffers for a file if more than one record at a time is needed from that file. These buffers can be SHARED among procedures.

```
DEFINE [ [ NEW ] SHARED ] BUFFER buffer FOR file [ PRESELECT ]
```

DEFINE PARAMETER *Statement*

Defines a runtime parameter in a called procedure. Each parameter requires one DEFINE statement, and definitions must match the order in which they are passed in a RUN statement.

$$\text{DEFINE } \left\{ \begin{array}{l} \text{INPUT} \\ \text{OUTPUT} \\ \text{INPUT-OUTPUT} \end{array} \right\} \text{ PARAMETER } \textit{parameter}$$

$$\left\{ \begin{array}{l} \text{AS } \textit{datatype} \\ \text{LIKE } \textit{field} \end{array} \right\} \left[\begin{array}{l} \text{COLUMN-LABEL } \textit{label} [! \textit{label}] \dots \\ \text{DECIMALS } \textit{n} \\ \text{FORMAT } \textit{string} \\ \text{INITIAL } \{ [\textit{constant} \dots] \} \\ \text{LABEL } \textit{string} \\ \text{NO-UNDO} \end{array} \right] \dots$$

DEFINE SHARED FRAME *Statement*

Defines a frame for use within a procedure or within several procedures.

DEFINE [NEW] SHARED FRAME *frame*

DEFINE STREAM *Statement*

Defines a stream other than the two unnamed streams (the input and output stream) which are automatically available.

DEFINE [[NEW
NEW GLOBAL] SHARED] STREAM *stream*

DEFINE VARIABLE *Statement*

Defines a variable (a temporary field) for use within a procedure or within several procedures.

```

DEFINE [ [ NEW
        NEW GLOBAL ] SHARED ] VARIABLE variable
      { AS datatype }
      { LIKE field }
      [ [NOT] CASE-SENSITIVE
        COLUMN-LABEL label [!label]...
        DECIMALS n
        EXTENT n
        FORMAT string
        INITIAL { [ constant ,... ] }
        LABEL string
        NO-UNDO
      ] ...
  
```


DEFINE WORKFILE STATEMENT

Defines a work file (a temporary database file) for use within a procedure or within several procedures.

```
DEFINE [ [ NEW ] SHARED ] WORKFILE workfile-name [ NO-UNDO ]  
      [ LIKE file-name ]
```

```
[ FIELD field-name  
  { AS datatype  
    LIKE field }  
  [ [NOT] CASE-SENSITIVE  
    COLUMN-LABEL label [ !label ] ...  
    DECIMALS n  
    EXTENT n  
    FORMAT string  
    INITIAL { [ constant, ... ] }  
    LABEL string ] ] ...
```

DELETE Statement

Removes a record from a record buffer and from the database.

DELETE ALIAS Statement

Deletes an alias from the alias table.

```
DELETE record [ VALIDATE (condition, msg-expression) ]
```

```
DELETE ALIAS alias
```

DELETE FROM (SQL) Statement

Deletes one or more rows from a table.

```
DELETE FROM table-name [ WHERE { search-condition |  
{ CURRENT OF cursor-name } } ]
```

DICTIONARY *Statement*

Runs the PROGRESS Data
Dictionary.

DICTIONARY

DISCONNECT *Statement*

Disconnects the specified database.

DISCONNECT *logical-name*

DISPLAY Statement

Moves data to a screen buffer,
displaying that data on the screen
or at a designated output destination.

DISPLAY [STREAM *stream*]

| | | |
|--|-----|-----|
| <p><i>expression</i> [<i>format-phrase</i>] [WHEN <i>expression</i>] [(<i>aggregate-phrase</i>)]</p> | ... | ... |
| <p><i>expression</i> [FORMAT <i>string</i> WHEN <i>expression</i>] ... @<i>base-field</i></p> | | |
| <p>[<i>format-phrase</i>]</p> | | |
| <p>SPACE [(<i>n</i>)]</p> | | |
| <p>SKIP[(<i>n</i>)]</p> | | |
| <p>[<i>frame-phrase</i>]</p> | | |

DISPLAY [STREAM *stream*] *record* [EXCEPT *field...*] [*frame-phrase*]

DO Statement

Groups statements into a single block, optionally specifying various processing services, or block properties.

```
[ label: ] DO  
[ FORrecord [ , record ] ... ]  
[ PRESELECT [ EACH ] record-phrase  
  [ , [ EACH ] record-phrase ] ...  
  [ [ BREAK ] { BY expression [ DESCENDING } ... ] ] ]  
[ variable = expression1 TO expression2 [ BY k ] ]  
[ WHILE expression ]  
[ TRANSACTION ]  
[ ON ENDKEY-phrase ]  
[ ON ERROR-phrase ]  
[ frame-phrase ]
```

DOS Statement

Runs a program, DOS command, or DOS batch file, or starts the DOS command processor, allowing interactive processing of DOS commands. A procedure containing the DOS statement will run on a UNIX system only if flow of control does not pass through that DOS statement.

$$\text{DOS [SILENT] } \left[\begin{array}{l} \textit{dos-command} \\ \text{VALUE(expression) } \end{array} \left[\begin{array}{l} \textit{argument} \\ \text{VALUE(expression) } \end{array} \right] \dots \right]$$
DOWN Statement

Explicitly positions to a new line in a Down or multi-line frame. Moves down if the expression is positive, up if the expression is negative.

$$\text{DOWN [STREAM } \textit{stream} \text{] [expression] [frame-phrase]}$$

DROP INDEX (SQL) Statement

Removes an index.

DROP INDEX *index-name*

DROP TABLE (SQL) Statement

Removes a table from the database. It also removes all indexes defined on that table and all access privileges, as well as all data.

DROP TABLE *table-name*

DROP VIEW (SQL) Statement

Removes a view from the database.

DROP VIEW *view-name*

EDITING *Phrase*

Identifies the processing to take place following each keystroke during a PROMPT-FOR, SET, or UPDATE statement.

[*label* :] EDITING: *statement*... END.

ENCODE *Function*

Encodes a source string to an encoded string.

ENCODE (*expression*)

END *Statement*

Indicates the end of a block started with a DO, FOR EACH, or REPEAT statement, or with an EDITING Phrase.

END

ENTERED *Function*

Returns a TRUE value if a frame field was modified during the last INSERT, PROMPT-FOR, SET, or UPDATE statement which used the field.

[FRAME *frame*] *field* ENTERED

ENTRY *Function*

Returns a character string entry from a list, based on a specified integer position. Separate multiple entries in the list with commas. ENTRY raises the ERROR condition if the value of the element does not correspond to the entry.

ENTRY (*element*, *list*)

EQ or = *Operator*

Returns a TRUE value if two expressions are equal.

expression { EQ } *expression*
 =

EXP Function

Returns a value resulting from raising a number (base) to a power (exponent).

EXP(*base,exponent*)

EXPORT Statement

Converts data to a standard character format for display to an output destination.

EXPORT [STREAM *stream*] { *expression* ...
record [EXCEPT *field* ...] }

FETCH (SQL) Statement

Retrieves the next row from the retrieval set accessed by the OPEN statement.

FETCH *cursor-name* INTO *variable-list*

FILL *Function*

Generates a character string made up of a character string expression repeated a specified number of times.

FILL (*expression, repeats*)

FIND *Statement*

Uses an index to locate a single record in a file and moves the record into a record buffer.

FIND $\left[\begin{array}{c} \text{FIRST} \\ \text{LAST} \\ \text{NEXT} \\ \text{PREV} \end{array} \right]$ *record-phrase* [NO-WAIT] [NO-ERROR]

FIRST *Function*

Returns a TRUE value if the current iteration of a DO, FOR EACH, or REPEAT ... BREAK block is the first iteration of that block.

FIRST(*break-group*)

FIRST-OF *Function*

Returns a TRUE value if the current iteration of a DO, FOR EACH, or REPEAT ... BREAK block is the first iteration for a new *break-group*.

FIRST-OF (*break-group*)

FOR Statement

Starts an iterating block that, at the start of each block iteration, reads a record from each of one or more files into record buffers.

[*label:*] FOR [EACH
FIRST
LAST] *record-phrase* [, [EACH
FIRST
LAST] *record-phrase*] ...

[[BREAK] { BY *expression* [DESCENDING] } ...]

[*variable* = *expression1* TO *expression2* [BY *k*]]

[WHILE *expression*]

[TRANSACTION]

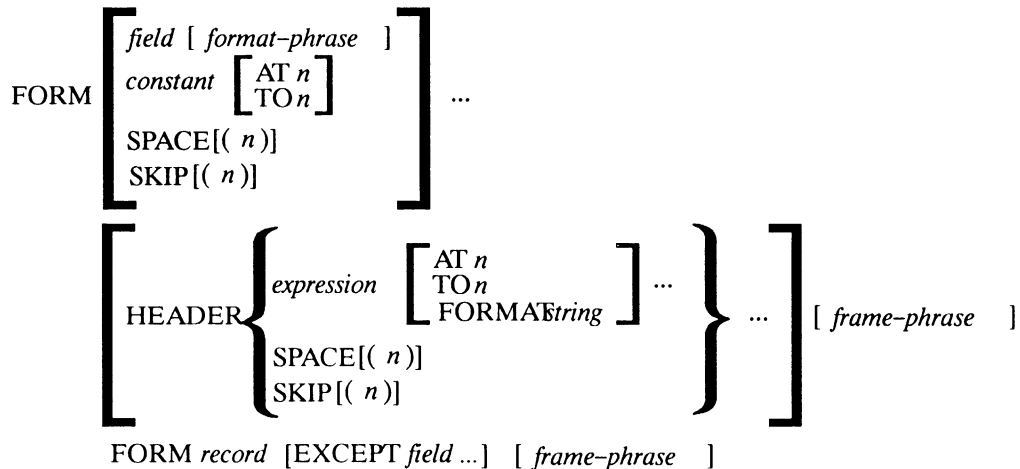
[ON ENDKEY-*phrase*]

[ON ERROR-*phrase*]

[*frame-phrase*]

FORM Statement

Defines the layout and certain processing attributes of a frame.



FORMAT *Phrase*

Specifies one or more frame attributes for a field, variable, or expression.

[
AT *n*
AS *datatype*
ATTR-SPACE
AUTO-RETURN
BLANK
COLON *n*
COLUMN-LABEL *label* [! *label*]...
DEBLANK
FORMAT *string*
HELP *string*
LABEL *string*
LIKE *field*
NO-ATTR-SPACE
NO-LABEL
TO *n*
VALIDATE (*condition*, *msg-expression*)
]

...

FRAME Phrase

Specifies the overall layout or processing properties of a frame.

When used on block header statements (DO, FOR EACH, and REPEAT), the Frame phrase also specifies the default frame for data handling statements (DISPLAY, SET, etc) within the block. Frame phrases can also be used on individual statements to indicate the specific frame to which the statement applies.

WITH

ACCUM
ATTR-SPACE
CENTERED

COLOR { [DISPLAY] *color-phrase* } ...
 PROMPT *color-phrase* }

COLUMN *expression*

n COLUMNS

DOWN

expression DOWN

FRAME *frame*

NO-ATTR-SPACE

NO-BOX

NO-HIDE

NO-LABELS

NO-UNDERLINE

NO-VALIDATE

OVERLAY

PAGE-BOTTOM

PAGE-TOP

RETAIN *n*

ROW *expression*

SCROLL *n*

SIDE-LABELS

TITLE [COLOR *color-phrase*] *expression*

TOP-ONLY

WIDTH *n*

...

FRAME-COL *Function*

Returns an integer value that represents the column position of the upper left corner of a frame.

FRAME-DB *Function*

Returns the database name for the field in which the cursor was last positioned for input.

FRAME-DOWN *Function*

Returns an integer value that represents the number of iterations in a frame.

FRAME-FIELD *Function*

During a data entry statement, returns the name of the input field the cursor is in. At other times, returns the name of the input field the cursor was last in.

FRAME-COL [(*frame*)]**FRAME-DB****FRAME-DOWN** [(*frame*)]**FRAME-FIELD**

FRAME-FILE *Function*

During a data entry statement, returns the name of the file containing the field the cursor is in. At other times, returns the name of the file containing the field the cursor was last in.

FRAME-INDEX *Function*

During a data entry statement, returns the subscript of the array element of the input field to which the cursor is currently positioned. At other times, returns the subscript of the array element to which the cursor was last positioned.

FRAME-LINE *Function*

Returns an integer value that represents the current logical line number in a down frame.

FRAME-FILE

FRAME-INDEX

FRAME-LINE [(*frame*)]

FRAME-NAME *Function*

Returns the name of the frame, if the cursor was last positioned to a field that is enabled for input.

FRAME-ROW *Function*

Returns an integer value that represents the row position of the upperleft corner of a frame.

FRAME-VALUE *Function*

During a data entry statement, returns the value of the input field the cursor is in. At other times, returns the value of the input field the cursor was last in.

FRAME-VALUE *Statement*

During a data entry statement, stores the value of an expression in a frame field.

FRAME-NAME**FRAME-ROW** [(*frame*)]**FRAME-VALUE****FRAME-VALUE** = *expression*

GATEWAYS *Function*

Returns a string containing a list of database types supported by the PROGRESS product from which it is executed.

GE or **> =** *Operator*

Returns a TRUE value if the first of two expressions is greater than or equal to the second expression.

GETBYTE *Function*

(RMS, Rdb, and ORACLE only)

Returns the integer value of the specified byte.

GATEWAYS
$$expression \left\{ \begin{array}{l} GE \\ > = \end{array} \right\} expression$$
GETBYTE(*expression,position*)

GO-PENDING *Function*

Returns a TRUE value if, within an EDITING Phrase, an APPLY statement has resulted in a GO action. The GO action is deferred until the end of the EDITING Phrase.

GRANT (*SQL Statement*)

Allows the owner or any user who holds the GRANT OPTION on a table or view to grant privileges on that table or view.

GO-PENDING

GRANT

```
{ ALL [ PRIVILEGES ] |  
  { SELECT |  
    INSERT |  
    DELETE |  
    { UPDATE [(column-list)] } [...] } }  
ON table-name TO { grantee-list | PUBLIC } [WITH GRANT OPTION]
```

GT or **>** *Operator*

Returns a TRUE value if the first of two expressions is greater than the second.

$$\textit{expression} \left\{ \begin{array}{c} \text{GT} \\ > \end{array} \right\} \textit{expression}$$
HIDE *Statement*

Removes a frame from the terminal screen, or clears the message area, or clears all frames and messages.

$$\text{HIDE [STREAM } \textit{stream}] \left[\begin{array}{l} \text{FRAME } \textit{frame} \\ \text{MESSAGE} \\ \text{ALL} \end{array} \right] [\text{NO-PAUSE}]$$
IF ... THEN ... ELSE *Function*

Evaluates one of two expressions depending on the value of a specified condition.

$$\text{IF } \textit{condition} \text{ THEN } \textit{expression1} \text{ ELSE } \textit{expression2}$$

IF ... THEN ... ELSE *Statement*

Makes execution of a statement or a block of statements conditional. If the value of the expression following the IF statement is TRUE, PROGRESS processes the statement following the THEN statement. Otherwise, PROGRESS processes the statements following the ELSE statement.

$$\text{IF } \textit{expression} \text{ THEN } \left\{ \textit{block statement} \right\} \left[\text{ELSE } \left\{ \textit{block statement} \right\} \right]$$

IMPORT *Statement*

The IMPORT Statement is the counterpart of the EXPORT statement. It reads a line from an input file that, typically, has been created by EXPORT. No format restrictions apply.

$$\text{IMPORT } [\text{STREAM } \textit{stream}] \left\{ \begin{array}{l} \textit{fields} \\ \textit{record} [\text{EXCEPT } \textit{field} \dots] \\ [\wedge] \end{array} \right\}$$

INDEX *Function*

Returns an integer value indicating the position of the target string within the source string.

INDEX(*source*, *target*)**INPUT** *Function*

References the value of a field in a screen buffer (*frame*).

INPUT [**FRAME** *frame*] *field***INPUT CLEAR** *Statement*

Clears any keystrokes buffered from the keyboard, discarding any “type-ahead” characters.

INPUT CLEAR**INPUT CLOSE** *Statement*

Closes the default input source or the stream you name.

INPUT [**STREAM** *stream*] **CLOSE**

INPUT FROM *Statement*

Specifies a new input source.

INPUT [STREAM *stream*] FROM

{ *opsys-file*
opsys-device
TERMINAL
VALUE(*expression*) }

[ECHO
NO-ECHO
UNBUFFERED
MAP *protermcap-entry*
[NO-] MAP] ...

INPUT THROUGH *Statement*

Uses the output from a UNIX program as the input to a PROGRESS procedure.

INPUT [STREAM *stream*] THROUGH

{ *program-name*
VALUE(*expression*) }

[*argument*
VALUE(*expression*)] ...

[ECHO
NO-ECHO
UNBUFFERED
MAP *protermcap-entry*
[NO-] MAP] ...

INPUT-OUTPUT CLOSE *Statement*

Closes a specified or default stream opened by an INPUT-OUTPUT THROUGH statement.

INPUT-OUTPUT [STREAM *stream*] CLOSE

INPUT-OUTPUT THROUGH *Statement*

Names a UNIX program (process) that PROGRESS will start. This process becomes the input source as well as the output destination for the procedure. A procedure containing the INPUT-OUTPUT THROUGH statement will run on a DOS or VMS machine only if flow of control does not pass through that statement.

INPUT-OUTPUT [STREAM *stream*] THROUGH

{ *program-name*
VALUE(*expression*) } [*argument*
VALUE(*expression*)] ... [ECHO
NO-ECHO
UNBUFFERED
MAP *protermcap-entry*
[NO-] MAP] ...

INSERT *Statement*

Creates a new database record, displays the initial values for the fields in the record, prompts for values of those fields, and assigns those values to the record.

INSERT *record* [EXCEPT *field ...*] [*frame-phrase*] [USING RECID (*n*)]

INSERT INTO *(SQL) Statement*

Adds new rows to a table.

INSERT INTO *table-name* [(*column-list*)] { **VALUES** (*value-list*) | **SELECT-statement** }

INTEGER *Function*

Converts an expression of any data type to an integer value, rounding that value if necessary.

INTEGER(*expression*)

IS-ATTR-SPACE *Function*

Returns a logical value that indicates if the current terminal type is spacetaking or non-spacetaking. Returns a value of Yes if the terminal is spacetaking, or a value of No if the terminal is non-spacetaking.

IS-ATTR-SPACE

KBLABEL *Function*

Returns the keyboard label (such as F1) of the primary key that performs a specified PROGRESS function (such as GO). See “The PROGRESS Keyboard” section for a list of key functions and the corresponding standard keyboard keys.

KEYCODE *Function*

Evaluates the key-label (such as F1) for a key in the predefined set of keyboard keys and returns the corresponding integer key code (such as 301). See “The PROGRESS Keyboard” section for a list of key codes and key-labels.

KBLABEL (*key-function*)**KEYCODE** (*key-label*)

KEYFUNCTION *Function*

Evaluates an integer expression (such as 301) and returns a character string that is the function of the key associated with that integer expression (such as GO). See “The PROGRESS Keyboard” section for a list of key functions.

KEYLABEL *Function*

Evaluates a key-code (such as 301) and returns a character string that is the predefined keyboard label for that key (such as F1). See “The PROGRESS Keyboard” section for a list of key codes and key labels.

KEYWORD *Function*

Returns a character value indicating whether the supplied string is a PROGRESS keyword.

KEYFUNCTION(*expression*)**KEYLABEL** (*key-code*)**KEYWORD** (*expression*)

LAST Function

Returns a TRUE value if the current iteration of a DO, FOR EACH, or REPEAT ... BREAK block is the last iteration of that block.

LASTKEY Function

Returns the integer key code of the most recent key pressed during an interaction with a procedure.

LAST-OF Function

Returns a TRUE value if the current iteration of a DO, FOR EACH, or REPEAT ... BREAK block is the last iteration for a particular value of a break group.

LAST (break-group)

LASTKEY

LAST-OF (break-group)

LC Function

Returns a character string identical to a specified string, but with any uppercase letters in that string converted to lowercase.

LC (*string*)

LDBNAME Function

Returns the logical name of a currently connected database.

LDBNAME { *(integer-expression)*
(logical-name)
(alias) }

LE or <= Operator

Returns a TRUE value if the first of two expressions is less than or equal to the second.

expression { LE
<= } *expression*

LEAVE *Statement*

Exits from a block. Execution continues with the first statement after the end of the block.

LEAVE [*label*]

LENGTH *Function*

Returns the length of a character string.

LENGTH (*string*)

LENGTH *Function*

(RMS, Rdb, and ORACLE only)

Returns the number of bytes in a raw datatype expression.

LENGTH (*expression*)

LENGTH *Statement*

(RMS, Rdb, and ORACLE only)

Changes the number of bytes in a raw data type variable.

LENGTH (*variable*) = *integer expression*

LIBRARY *Function*

Parses a character string in the form *path-name* << *member-name* >>, where *path-name* is the pathname of a library and *member-name* is the name of a file within the library, and returns the name of the library. The brackets << >> indicate that *member-name* is a file in a library. If the string is not in this form, the **LIBRARY** function returns an unknown value (?).

LIBRARY (*string*)

LINE-COUNTER *Function*

Returns the current line number of paged output. Returns zero if the current output is not paged.

LINE-COUNTER[(*stream*)]

LOCKED *Function*

Returns a TRUE value if, because another user has locked a record, that record was not available to a prior FIND...NO-WAIT statement.

LOCKED *record*

LOG *Function*

Calculates the logarithm of an expression using a specified base.

$\text{LOG}(\textit{expression} [,\textit{base}])$

LOOKUP *Function*

Returns an integer giving the position of a character expression in a list. Each entry in the list is separated from the next by a comma. Returns 0 if the expression is not found.

$\text{LOOKUP}(\textit{expression},\textit{list})$

LT or **<** *Operator*

Returns a TRUE value if the first of two expressions is less than the second.

$\textit{expression} \left\{ \begin{array}{c} \text{LT} \\ < \end{array} \right\} \textit{expression}$

MATCHES *Function*

Compares a character expression to a pattern and returns a TRUE value if the expression satisfies the pattern criteria. Use an asterisk (*) in the pattern to match 0 or more characters and a period (.) to match exactly one character.

expression MATCHES *pattern*

MAXIMUM *Function*

Compares two values (expressions) and returns the larger of the two values.

MAXIMUM(*expression1*, *expression2*)

MEMBER *Function*

Parses a character string in the form *path-name*<<*member-name*>>, where *path-name* is the pathname of a library and *member-name* is the name of a file within the library, and returns *member-name*. The brackets << >> indicate that *member-name* is a file in a library. If the string is not in this form, the MEMBER function returns an unknown value (?).

MESSAGE *Statement*

Displays messages in the message area at the bottom of the terminal screen.

MEMBER(*string*)

MESSAGE [**COLOR** *color-phrase*] [*expression*] ...

[{ **SET**
UPDATE } *field* [**FORMAT** *string*
AUTO-RETURN]]

MESSAGE-LINES *Function*

Returns the number of lines in the message area at the bottom of the terminal screen. (Always returns 2).

MINIMUM *Function*

Compares two values (expressions) and returns the smaller of the two values.

MESSAGE-LINES

MINIMUM(*expression1*, *expression2*)

MODULO *Function*

Returns the remainder after division.

expression MODULO *base*

MONTH *Function*

Returns the month value (from 1 to 12) of a date you specify.

MONTH (*date*)

NE or **< >** *Operator*

Compares two expressions and returns a TRUE value if they are not equal.

expression { **NE** } *expression*
< >

NEW *Function*

Checks a record buffer and returns a TRUE value if the record in that buffer was newly created. If the record was read from the database, NEW returns a FALSE value.

NEW *record*

NEXT Statement

Goes directly to the END of an iterating block and starts the next iteration of the block.

NEXT [*label*]

NEXT-PROMPT Statement

The NEXT-PROMPT statement specifies which field to first position the cursor on during the next input operation involving that frame field.

NEXT-PROMPT *field* [*frame-phrase*]

NOT Function

Return TRUE if an expression is false and FALSE if an expression is true.

NOT *expression*

NOT ENTERED Function

Returns a TRUE value if a frame field was not modified during the last INSERT, PROMPT-FOR, SET, or UPDATE statement which used the field.

field **NOT ENTERED**

NUM-ALIASES *Function*

Returns an integer value representing the number of aliases defined. The NUM-ALIASES function takes no arguments.

NUM-DBS *Function*

Returns the number of connected databases.

NUM-ENTRIES *Function*

Returns the number of items in a comma-separated list of strings. See also the ENTRY function.

NUM-ALIASES

NUM-DBS

NUM-ENTRIES (*string-expression*)

ON Statement

Indicates the action to be taken when the user presses a special key (such as F1) in response to an INSERT, PROMPT-FOR, SET, or UPDATE statement, or when the procedure pauses (either PROGRESS has encountered a PAUSE statement or is pausing because the screen is full). If the input request comes from a READKEY statement, the action specified by the ON statement is not taken. See "The PROGRESS Keyboard" section for a list of key labels and key functions.

ON *key-label key-function*

ON ENDKEY Phrase

Describes the processing that takes place when the ENDKEY condition occurs during a block. This condition usually occurs when a user presses the keyboard ENDKEY during the first interaction of a block iteration.

ON ERROR Phrase

Describes the processing that takes place when there is an error during a block.

OPEN (SQL) Statement

Selects the retrieval set from the execution of the SELECT clause in a DECLARE CURSOR statement.

ON ENDKEY UNDO[*label1*] [, LEAVE [*label2*]
 , NEXT [*label2*]
 , RETRY [*label2*]
 , RETURN]

ON ERROR UNDO[*label1*] [, LEAVE [*label2*]
 , NEXT [*label2*]
 , RETRY [*label2*]
 , RETURN]

OPEN *cursor-name*

OPSYS *Function*

Returns a value of MSDOS, UNIX, OS2, or VMS, depending on the operating system on which you are running PROGRESS.

OR *Function*

Returns a TRUE value if either of two logical expressions is TRUE.

OS2 *Statement*

Runs a program, OS/2 command, OS/2 batch file, or starts the OS/2 command processor, allowing interactive processing of OS/2 commands.

OPSYS

expression OR *expression*

OS2 [SILENT] [*os2-command*
VALUE (*expression*) [*argument*
VALUE (*expression*)] ...]

OUTPUT CLOSE *Statement*

Closes the default output destination or the output stream you name with the STREAM keyword.

OUTPUT [STREAM *stream*] CLOSE

OUTPUT THROUGH *Statement*

Identifies a new output destination as the input to a UNIX process that PROGRESS will start. A procedure containing the OUTPUT THROUGH statement will run on a DOS or VMS system only if flow of control does not pass through that statement.

OUTPUT [STREAM *stream*] THROUGH

{ *program-name*
VALUE(*expression*) }

[*argument*
VALUE(*expression*)] ...

[PAGED
PAGE-SIZE *n*
ECHO
NO-ECHO
UNBUFFERED
[NO-] MAP] ...

OUTPUT TO*Statement*

Specifies a new output destination.

OUTPUT [STREAM*stream*] TO

| | | |
|---|---|-----|
| $\left\{ \begin{array}{l} \text{PRINTER} \\ \text{opsys-file} \\ \text{opsys-device} \\ \text{TERMINAL} \\ \text{VALUE(expression)} \end{array} \right\}$ | $\left[\begin{array}{l} \text{PAGED} \\ \text{PAGE-SIZE } \textit{expression} \\ \text{APPEND} \\ \text{ECHO} \\ \text{NO-ECHO} \\ \text{UNBUFFERED} \\ \text{[NO-] MAP } \textit{protermcap-entry} \end{array} \right]$ | ... |
|---|---|-----|

OVERLAY *Function*

Overlays a character expression in a field or variable starting at a given position, and optionally for a given length.

OVERLAY (*target*, *position* [,*length*]) = *expression*

PAGE *Statement*

Starts a new output page for PAGED output. No action is taken if output is already positioned at the beginning of a page.

PAGE-NUMBER *Function*

Returns the page number of an output destination. If the output stream is not paged, returns a value of 0.

PAGE-SIZE *Function*

Returns the page size (lines per page) of an output destination. If the output stream is not paged, PAGE-SIZE returns a value of 0.

PAGE [**STREAM** *stream*]

PAGE-NUMBER [(*stream*)]

PAGE-SIZE [(*stream*)]

PAUSE *Statement*

Suspends processing indefinitely, or for a specified number of seconds, or until the user presses any key. Specifying BEFORE-HIDE controls the pause duration and message before frames are automatically hidden.

$$\text{PAUSE } [n] [\text{BEFORE-HIDE}] \left[\begin{array}{l} \text{MESSAGE } \textit{message} \\ \text{NO-MESSAGE} \end{array} \right]$$
PDBNAME *Function*

Returns the physical name of a currently connected database.

$$\text{PDBNAME} \left\{ \begin{array}{l} \textit{(integer-expression)} \\ \textit{(logical-name)} \\ \textit{(alias)} \end{array} \right\}$$
PROGRAM-NAME *Function*

Returns the name of the calling program.

$$\text{PROGRAM-NAME}(n)$$

PROGRESS *Function*

Returns one of the following character values identifying the PROGRESS product that is running: PROGRESS 4GL/RDBMS (Full), Query/Run-Time, Run-Time.

Can also return COMPILE if you are using the developer's toolkit, or COMPILE-ENCRYPT if you are using the run-timer compiler.

PROGRESS

PROMPT-FOR *Statement*

Requests input and places
that input in the screen buffer
(frame).

```

PROMPT-FOR [ STREAM $stream$  ]
    [
         $field$  [  $format-phrase$  ] [ WHEN  $expression$  ]
        TEXT ( $field$  [ $format-phrase$ ] ... )
         $constant$  [ AT  $n$ 
                    TO  $n$  ]
        ^
        SPACE[(  $n$  )]
        SKIP[(  $n$  )]
    ] ...
    [ GO-ON (  $key-label$  ... ) ]
    [  $frame-phrase$  ]
    [ EDITING- $phrase$  ]

PROMPT-FOR [ STREAM $stream$  ]  $record$  [ EXCEPT  $field$  ... ] [  $frame-phrase$  ]
  
```

PROPATH *Statement*

Sets the PROPATH environment variable for the current PROGRESS session only (new PROPATH not inherited by any subprocesses).

PROPATH *=string-expression*

PROPATH *Function*

Returns a comma-separated list of the directory paths in the PROPATH environment variable.

PROPATH

PUT Statement

Sends the value of one or more expressions to an output destination other than the terminal.

PUT [STREAM *stream*] [UNFORMATTED]

| | | | | | | | | | | | | | |
|---|--|--|--------------------------|----------------------|----------------------|----------------------|----------------------|--|-----------------------------|--|------------------------------|--|-----|
| <table style="border-collapse: collapse; margin-left: auto; margin-right: auto;"> <tr> <td style="padding-right: 10px;"><i>expression</i></td> <td style="border-left: 2px solid black; border-right: 2px solid black; padding: 10px 10px 10px 10px;"> <table style="border-collapse: collapse; margin-left: auto; margin-right: auto;"> <tr> <td style="padding-right: 10px;">FORMAT <i>expression</i></td> <td style="border-left: 2px solid black; border-right: 2px solid black; padding: 10px 10px 10px 10px;">...</td> </tr> <tr> <td style="padding-right: 10px;">AT <i>expression</i></td> <td style="border-left: 2px solid black; border-right: 2px solid black; padding: 10px 10px 10px 10px;"></td> </tr> <tr> <td style="padding-right: 10px;">TO <i>expression</i></td> <td style="border-left: 2px solid black; border-right: 2px solid black; padding: 10px 10px 10px 10px;"></td> </tr> </table> </td> </tr> <tr> <td style="padding-right: 10px;">SKIP[(<i>expression</i>)]</td> <td style="border-left: 2px solid black; border-right: 2px solid black; padding: 10px 10px 10px 10px;"></td> </tr> <tr> <td style="padding-right: 10px;">SPACE[(<i>expression</i>)]</td> <td style="border-left: 2px solid black; border-right: 2px solid black; padding: 10px 10px 10px 10px;"></td> </tr> </table> | <i>expression</i> | <table style="border-collapse: collapse; margin-left: auto; margin-right: auto;"> <tr> <td style="padding-right: 10px;">FORMAT <i>expression</i></td> <td style="border-left: 2px solid black; border-right: 2px solid black; padding: 10px 10px 10px 10px;">...</td> </tr> <tr> <td style="padding-right: 10px;">AT <i>expression</i></td> <td style="border-left: 2px solid black; border-right: 2px solid black; padding: 10px 10px 10px 10px;"></td> </tr> <tr> <td style="padding-right: 10px;">TO <i>expression</i></td> <td style="border-left: 2px solid black; border-right: 2px solid black; padding: 10px 10px 10px 10px;"></td> </tr> </table> | FORMAT <i>expression</i> | ... | AT <i>expression</i> | | TO <i>expression</i> | | SKIP[(<i>expression</i>)] | | SPACE[(<i>expression</i>)] | | ... |
| <i>expression</i> | <table style="border-collapse: collapse; margin-left: auto; margin-right: auto;"> <tr> <td style="padding-right: 10px;">FORMAT <i>expression</i></td> <td style="border-left: 2px solid black; border-right: 2px solid black; padding: 10px 10px 10px 10px;">...</td> </tr> <tr> <td style="padding-right: 10px;">AT <i>expression</i></td> <td style="border-left: 2px solid black; border-right: 2px solid black; padding: 10px 10px 10px 10px;"></td> </tr> <tr> <td style="padding-right: 10px;">TO <i>expression</i></td> <td style="border-left: 2px solid black; border-right: 2px solid black; padding: 10px 10px 10px 10px;"></td> </tr> </table> | FORMAT <i>expression</i> | ... | AT <i>expression</i> | | TO <i>expression</i> | | | | | | | |
| FORMAT <i>expression</i> | ... | | | | | | | | | | | | |
| AT <i>expression</i> | | | | | | | | | | | | | |
| TO <i>expression</i> | | | | | | | | | | | | | |
| SKIP[(<i>expression</i>)] | | | | | | | | | | | | | |
| SPACE[(<i>expression</i>)] | | | | | | | | | | | | | |

PUT [STREAM *stream*] CONTROL *expression* ...

PUT CURSOR *Statement*

Makes the cursor visible on the screen.

PUT CURSOR { OFF
{ ROW *expression*
COLUMN *expression* } ... }

PUTBYTE *Function*

(RMS, Rdb, and ORACLE only)

Replaces a byte in a variable with the integer value of an expression.

PUTBYTE(*variable,position*) = *expression*

PUT SCREEN *Statement*

Displays a character expression at a specified location on a screen, overlaying any other data that might be displayed at that location.

PUT SCREEN [ATTR-SPACE
COLOR*color-phrase*
COLUMN*expression*
NO-ATTR-SPACE
ROW *expression*] ... *expression*

QUIT *Statement*

Exits from PROGRESS and returns to the operating system.

R-INDEX *Function*

Returns an integer that indicates the position of the target string within the source string. In contrast to the index function, the search is performed from right to left.

RANDOM *Function*

Returns a random integer between two integers (inclusive).

QUIT

R-INDEX (*source, target*)

RANDOM(*low, high*)

RAW Function

(RMS, Rdb, and ORACLE only)

Extracts bytes from a field.

RAW Statement

(RMS, Rdb, and ORACLE only)

Writes bytes to a field.

READKEY Statement

Reads one keystroke from an input source and sets the value of LASTKEY to the keycode of that keystroke. See “The PROGRESS Keyboard” section for a list of key codes.

`RAW(field[,position[,length]])`

`RAW(field[,position[,length]]) = expression`

`READKEY [STREAM stream] [PAUSE n]`

RECID *Function*

Returns the unique internal identifier of the database record currently associated with the record buffer you name.

RECID(*record*)

Record *Phrase*

Identifies the record to retrieve with a FIND statement, the set of records to retrieve using a FOR EACH statement, or the constraints on the records being preselected in a DO or REPEAT block.

record [*constant*] [WHERE *expression*
 USING [FRAME *frame*] *field* [AND *field*] ...] ...
 OF *file*
 USE-INDEX *index*]
 [SHARE-LOCK
 EXCLUSIVE-LOCK
 NO-LOCK]

RELEASE *Statement*

Verifies that a record complies with mandatory field and unique index definitions, and clears the record from the buffer, writing it back to the database if it has been changed. Raises the ERROR condition if the validation fails.

RELEASE *record*

REPEAT Statement

Begins a block that has all of the automatic properties except record reading.

```
[ label: ] REPEAT
  [ FOR record [ , record ] ... ]
  [ PRESELECT[ EACH ]record-phrase
    [ , [ EACH ] record-phrase ] ...
    [ [ BREAK ] { BY expression [ DESCENDING ] } ... ]
  [ variable = expression1 TO expression2 [ BY k ] ]
  [ WHILE expression ]
  [ TRANSACTION ]
  [ ON ENDKEY-phrase ]
  [ ON ERROR-phrase ]
  [ frame-phrase ] ]
```

RETRY *Function*

Returns a TRUE value if the current block is being reprocessed after a previous UNDO, RETRY.

RETRY**RETURN** *Statement*

Leaves the procedure block, returning to the calling procedure or, if there there was no calling procedure, to the PROGRESS editor.

RETURN**REVOKE** *(SQL) Statement*

Allows the owner or any user who holds the GRANT OPTION on a table or view to revoke privileges on that table or view.

REVOKE

```
{ ALL [ PRIVILEGES | { SELECT | INSERT | DELETE | { UPDATE [(column-list)] }  
[,...] } }  
ON table-name FROM { grantee-list | PUBLIC }
```

ROLLBACK WORK (*SQL*) *Statement*

Discards all database changes effected by SQL data manipulation statements since the COMMIT WORK or previous ROLLBACK work statement or since the beginning of the session.

ROLLBACK WORK**ROUND** *Function*

Rounds a decimal expression to a specified number of places (precision) after the decimal point, returning a decimal value.

ROUND(*expression*, *precision*)

RUN *Statement*

Runs (calls) a PROGRESS procedure from within a procedure.

RUN { *procedure*
VALUE(*expression*) } [(*parameter* [, *parameter*])] [*argument*] ...

SCREEN-LINES *Function*

Returns the number of screen lines you can use to display frames. This value is three less than the total display lines available on the screen.

SCROLL *Statement*

Opens a space and moves data in a frame with multiple rows. Use the SCROLL Statement to scroll data up or down when you add or delete a line in a frame (often a scrolling frame).

SCREEN-LINES

SCROLL [FROM-CURRENT [**UP**
DOWN] [*frame-phrase*] ...

SDBNAME *Function*

For non-PROGRESS databases, the SDBNAME function returns the logical name of the schema holder database. For PROGRESS databases, it is equivalent to LDBNAME.

$$\text{SDBNAME} \left\{ \begin{array}{l} (\textit{integer-expression}) \\ (\textit{logical-name}) \\ (\textit{alias}) \end{array} \right\}$$
SEARCH *Function*

Searches the PROGRESS directory path (PROPATH) for a file. If the file is in your working directory, SEARCH returns the name of that file. If the file is not in your working directory, SEARCH returns the fully qualified path name for the file. If SEARCH does not find the file, it returns an unknown value (?).

SEARCH (*opsys-file*)

SEEK Function

Returns the offset (in bytes) of the file pointer in an ASCII file. You define a procedure variable to hold the offset value and later position the file to that offset.

$$\text{SEEK} \left(\left\{ \begin{array}{c} \text{INPUT} \\ \text{OUTPUT} \\ \textit{name} \end{array} \right\} \right)$$

SEEK Statement

Positions the file pointer to a user-defined offset (in bytes) to an ASCII file. This statement does not require that the file be closed and reopened.

$$\text{SEEK} \left\{ \begin{array}{c} \text{INPUT} \\ \text{OUTPUT} \\ \text{STREAM } \textit{stream} \end{array} \right\} \text{ TO } \left\{ \begin{array}{c} \textit{expression} \\ \text{END} \end{array} \right\}$$

SELECT (SQL) Statement

Retrieves and displays data from a table. Refer to Chapter 15 in the *Programming Handbook* for detailed information about the SELECT statement.

```
SELECT[ ALL | DISTINCT ] { * | column-list } [ INTO variable-list]  
    FROM { table-name [ range-variable ] } [...]  
    [ WHERE search-condition ]  
    [ GROUP BY column-list ]  
    [ HAVING search-condition ]  
    [ ORDER BY { { column-name | n } [ ASC | DESC ] } [...]]
```


SET Statement

Requests input, and then puts the input data in both the screen buffer (frame) and in the specified fields or variables.

```
SET [ STREAM stream ]  
  [ field [ format-phrase ] [ WHEN expression ]  
    TEXT(field [ format-phrase ] ...)  
    field = expression  
    constant [ AT n  
               ^  
               TO n ]  
    SPACE[( n )]  
    SKIP[( n ) ]  
  ] ...  
  [ GO-ON ( key-label ... ) ]  
  [ frame-phrase ]  
  [ EDITING phrase ]  
SET [ STREAM stream ] record [ EXCEPT field ... ] [ frame-phrase ]
```

SETUSERID *Function*

If the userid and password supplied to the SETUSERID function are in the _User file, SETUSERID returns a TRUE value and assigns the userid to the user. If the the userid is not in the _User file or the password is incorrect, SETUSERID returns a FALSE value and does not assign the userid to the user.

SQRT *Function*

Returns the square root (as a decimal value) of an expression. If a negative number is specified, SQRT returns an unknown value (?).

SETUSERID (*userid, password* [, *logical-dbname*])

SQRT (*expression*)

STATUS *Statement*

Specifies the text of the expression that appears on the bottom “status line” of the terminal screen.

$$\text{STATUS} \left\{ \begin{array}{l} \text{DEFAULT [expression]} \\ \text{INPUT [OFF} \\ \text{expression]} \end{array} \right\}$$

STOP *Statement*

Stops processing a procedure, backs out the active transaction, and returns to the start-up procedure or to the PROGRESS editor.

STOP

STRING *Function*

Converts a value of any data type into a character value. STRING uses EXPORT format if you do not supply a format. See the “Data Formats” section for information on format syntax.

STRING(*source* [, *format*])

SUBSTRING *Function*

Extracts a portion of a character string from a field or variable, or replaces characters in a field or variable with an expression and at a starting point you specify.

SUBSTRING *Statement*

Replaces characters in a field or variable with an expression you specify.

TERMINAL *Function*

Returns the value BW80, CO80, or MONO, depending on the monitor type, on DOS and OS/2 systems. Returns the value of the \$TERM variable on UNIX systems. Returns the value of the PROTERM variable on VMS systems. Returns null in batch mode on all systems.

SUBSTRING(*source*, *position* [, *length*])

SUBSTRING(*source*, *position* [, *length*]) = expression

TERMINAL

TERMINAL *Statement*

Changes terminal type during program execution. On UNIX and BTOS/CTOS systems, change the value of the TERM environment variable. On VMS systems, change the value of the PROTERM logical variable (or TERM if PROTERM has not been set).

TIME *Function*

Returns the number of seconds since midnight (local time). Used with the STRING function, the time can be formatted into hours, minutes, and seconds. See the "Data Formats" section for information on TIME format.

TODAY *Function*

Returns the current system date.

TERMINAL = *termid*

TIME

TODAY

TRIM *Function*

Removes leading and trailing spaces in a character string.

TRIM (*expression*)

TRUNCATE *Function*

Truncates a decimal expression to a specified number of places (*precision*), returning a decimal value.

TRUNCATE(*expression, precision*)

UNDERLINE *Statement*

Underlines a field or variable, using the next display line for the underline.

UNDERLINE [STREAM *stream*] *field* ... [*frame-phrase*]

UNDO *Statement*

Backs out all modifications to fields and variables made during the current iteration of a block, and indicates what action to take next.

UNDO[*label1*] [, LEAVE [*label2*]
 , NEXT [*label2*]
 , RETRY [*label2*]
 , RETURN]

UNIX Statement

Runs a program, UNIX command or UNIX script, or starts a UNIX interactive shell to allow interactive processing of UNIX commands.

UNIX [SILENT] [*unix-command* VALUE(*expression*) [*argument* VALUE(*expression*)] ...]

UP Statement

Explicitly positions to a new line in a down, or multi-line frame. Moves up if the expression is positive, down if the expression is negative.

UP [STREAM *stream*] [*expression*] [*frame-phrase*]

UPDATE Statement

Displays fields or variables, requests input, and then puts the input data in both the screen buffer (frame) and in the specified fields or variables.

UPDATE

| | | | |
|---|--|---|-----|
| [| <i>field</i> [<i>format-phrase</i>] [WHEN <i>expression</i>] TEXT(<i>field</i> [<i>format-phrase</i>] ...) <i>field</i> = <i>expression</i> <i>constant</i> [AT <i>n</i> TO <i>n</i>] ^ SPACE[(<i>n</i>)] SKIP[(<i>n</i>)] |] | ... |
|---|--|---|-----|

[GO-ON (*key-label* ...)][*frame-phrase*][EDITING-*phrase*]UPDATE *record* [EXCEPT *field* ...] [*frame-phrase*]

UPDATE *(SQL) Statement*

Changes values in one or more rows of a table.

```
UPDATE table-name
  SET column-name = expression [, column-name = expression] ...
  [ WHERE { search-condition | { CURRENT OF cursor-name } } ]
```

USERID *Function*

Returns the userid of the current user.

```
USERID [(logical-dbname)]
```

VALIDATE *Statement*

Verifies that a record complies with mandatory field and unique index definitions. Raises the ERROR condition if the validation fails.

```
VALIDATE record
```

VIEW Statement

Brings a frame into view or activates the frame for display at the beginning or end of a page if it is a PAGE-TOP or PAGE-BOTTOM frame.

VIEW [STREAM *stream*] [FRAME *frame*]

VMS Statement

Runs a program, VMS command or VMS command file, or starts an interactive VMS command processor.

VMS [SILENT] [ATTACH] [*vms-command*
VALUE(*expression*) [*argument*
VALUE(*expression*)] ...]

WEEKDAY *Function*

Evaluates a date expression and returns, as an integer, the day of the week from 1 (Sunday) to 7 (Saturday) for that date.

WEEKDAY(*date*)**YEAR** *Function*

Evaluates a date expression and returns the year value of that date, including the century.

YEAR(*date*)

OPERATOR PRECEDENCE TABLE

| Name of Operator | Precedence |
|---|-------------|
| - UNARY NEGATIVE + UNARY POSITIVE | 7 (highest) |
| MODULO / DIVISION * MULTIPLICATION | 6 |
| - DATE SUBTRACTION - SUBTRACTION + DATE ADDITION + CONCATENATION + ADDITION | 5 |

| Name of Operator | Precedence |
|---|------------|
| MATCHES LT or < LE or <= GT or > GE or >= EQ or = NE <> BEGINS | 4 |
| NOT | 3 |
| AND | 2 |
| OR | 1 (lowest) |

If an expression contains two operators of equal precedence, PROGRESS evaluates the expression from left to right. If the operators are not of equal precedence, PROGRESS evaluates the operator of higher precedence first. Use parentheses to change the default order used to evaluate an expression.

DATA FORMATS

| Data Type | Default Display Format | Syntax to Use When Specifying a Display Format | Notes |
|-----------|------------------------|---|--|
| Character | x(8) | $\left\{ \left\{ \begin{array}{c} X \\ N \\ A \\ ! \\ 9 \end{array} \right. \left[(n) \right] \right\} \dots$ <p style="text-align: center;"><i>fill character</i></p> | <p>Using an integer in parentheses represents a repetition factor for the previous non-fill character.</p> <p>To use X, N, A, ! or 9 as a fill character, you must precede that character with a tilde (~).</p> <p>To use (as a fill character after a non-fill character, you must precede it with atilde (~).</p> |
| Date | 99/99/99 | <p>9 ... /9 .../9 ...</p> <p>or 9 ...-9 ...-9 ...</p> | <p>Month/day/year order is assumed unless you used the -d parameter when starting PROGRESs.</p> |

DATA FORMATS (continued)

| Data Type | Default Display Format | Syntax to Use When Specifying a Display Format | Notes |
|-----------|------------------------|--|--|
| Decimal | ->>, >>9.99 | Numeric | See Numeric Format below. |
| Integer | ->, >>>, >>9 | Numeric | See Numeric Format below. |
| Logical | yes/no | [<i>string1</i>] [/ <i>string2</i>] | PROGRESS displays <i>string1</i> if the value is true, <i>string2</i> if false. Omitting a parameter defaults to blanks. |

NUMERIC FORMAT

| | |
|--|---|
| $[([string1] [+] [-] [(] [>] \dots [9] [Z] \dots [.] [9] [<] \dots [+] [-] [)] [string2]]$ | <p><i>String1</i> is any characters except:</p> <ul style="list-style-type: none"> plus + minus - greater than > comma , letter z (z or Z) digits 0 - 9 right parenthesis) left parenthesis (asterisk * period . <p><i>String2</i> has the same limitations as <i>String1</i>, except the period (.) is allowed.</p> |
|--|---|

TIME FORMAT
(Used with the STRING Function)

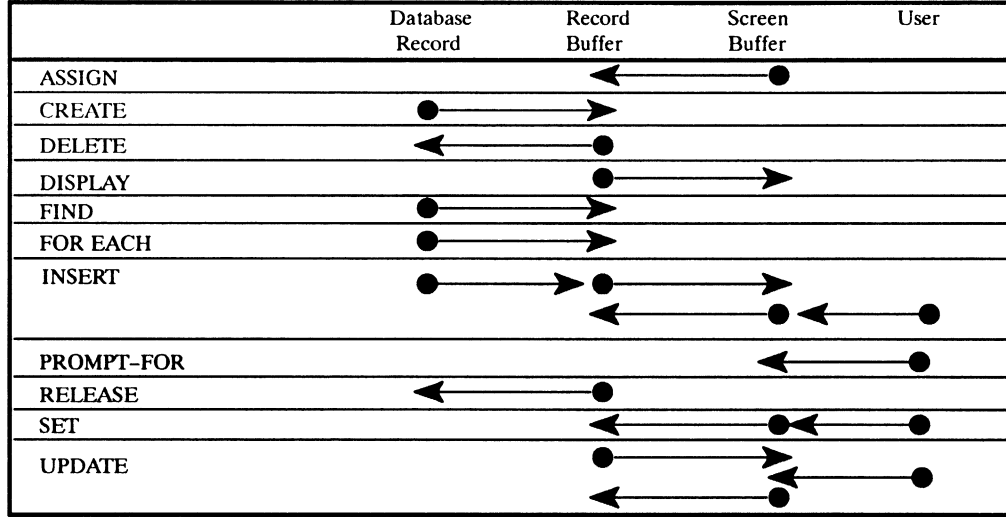
| | |
|------------------------------------|------------------------------------|
| HH:MM:SS [<i>any-characters</i>] | hh:mm:ss [<i>any-characters</i>] |
| HH:MM [<i>any-characters</i>] | hh:mm [<i>any-characters</i>] |

Uses 24-hour format unless *any-characters* contains an A or a.

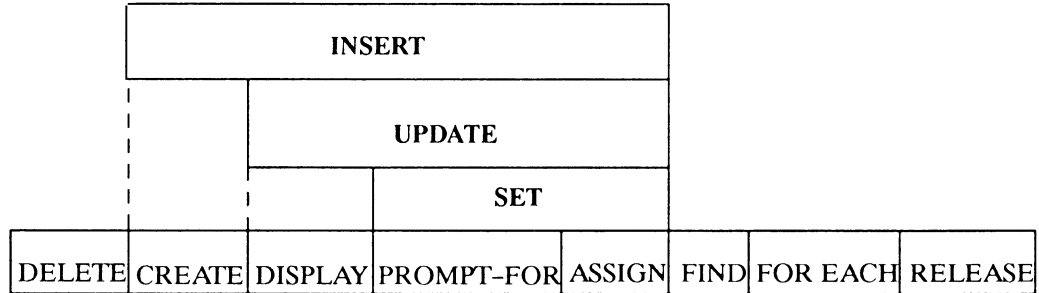
Example:

STRING(TIME, "HH:MM AM")

DATA HANDLING STATEMENTS AND DATA MOVEMENT



STATEMENTS AS BUILDING BLOCKS



BLOCK PROPERTIES

| PROPERTY | REPEAT | | FOR EACH | | DO | | Procedure | |
|---------------------|----------|----------------|----------|------------------|----------|---------------------------------|-----------|----------|
| | Implicit | Explicit | Implicit | Explicit | Implicit | Explicit | Implicit | Explicit |
| Looping | YES | WHILE TO/BY | YES | WHILE TO/BY | NO | WHILE TO/BY | NO | NO |
| Record reading | NO | NO | YES | RECORD Phrase | NO | NO | NO | NO |
| Frame scoping | YES | WITH FRAME | YES | WITH FRAME | NO | WITH FRAME | YES | NO |
| Record scoping | YES | FOR | YES | NO | NO | FOR | YES | NO |
| Record Preselection | NO | PRESELECT | NO | NO | NO | PRESELECT | NO | NO |
| UNDO | YES | NO | YES | NO | NO | TRANS- ACTION ON ERROR | YES | NO |

(continued)

BLOCK PROPERTIES

| PROPERTY | REPEAT | | FOR EACH | | DO | | Procedure | |
|--------------------|----------|--------------|----------|--------------|----------|------------------------|-----------|----------|
| | Implicit | Explicit | Implicit | Explicit | Implicit | Explicit | Implicit | Explicit |
| ERROR processing | YES | ON ERROR | YES | ON ERROR | NO | ON ERROR | YES | NO |
| ENDKEY processing | YES | ON ENDKEY | YES | ON ENDKEY | NO | ON ENDKEY | YES | NO |
| System Transaction | YES | TRANS-ACTION | YES | TRANS-ACTION | NO | TRANS-ACTION ON ERROR* | YES | NO |

(*) Only if DO block contains database updates or reads with exclusive locks.

PROGRESS MASTER INDEX

This master index is designed to help you locate information in the following PROGRESS manuals. The abbreviation following each manual title is the index references.

| | |
|--|-------|
| <i>PROGRESS Test Drive</i> | TD |
| <i>PROGRESS Language Tutorial</i> | TUT |
| <i>Programming Handbook</i> | HND |
| <i>PROGRESS Language Reference</i> | REF |
| <i>System Administration I: Environments</i> | SA I |
| <i>System Administration II: General</i> | SA II |
| <i>Using the Developer's Toolkit</i> | TK |
| <i>Database Gateways</i> | DG |
| <i>3GL Interface Manual</i> | 3GL |
| <i>Pocket PROGRESS</i> | PK |

A

- Activities file, HND 13
- Active database, TUT 3
- After-image file (.ai), SA II 1,7
- Aliases, HND 13
- ALIAS function, REF
- ALTER TABLE (*SQL*) statement, REF
- Aggregate phrase (TOTAL), TUT 12; REF
- AMBIGUOUS function, TUT 10; REF
- applhelp.p procedure, TUT 15; HND 3
- Application, TUT 1
 - databases, HND 3
 - debugging, HND 14
 - design, TUT 5, 7
 - distributing, TUT 16; TK 3; 8
 - implementation steps, TUT 16
 - packaging, TUT 16
 - multi-user, HND 12
 - security, HND 11, SA II 5
 - starting with scripts, TK 6
 - testing, HND 14; TK 8
 - transporting, HND 14, TK 7
 - upgrading, TK 9
 - user access, TUT 16
- APPLY statement, HND 6; REF
- Arrays, TUT 10; HND 3
- AS option, DEFINE VARIABLE statement, TUT 10; REF
- ASSIGN statement, TUT 8; REF
- ATTR-SPACE option, Frame phrase, HND 7 REF; TK 5
- Audit trails. *See* Transactions
- Auto-Connect, SA II 2
- AUTO-RETURN option
 - SET statement, TUT 11; REF
 - UPDATE statement, TUT 6; REF
- AVAILABLE function, TUT 10; REF

B

- Backing up a database, SA II 4
- Batch jobs, SA II 2,3
- Before-image file (.bi), HND 8; SA II 1,7
- Blank userid, HND 11

Blocks, TUT 4; HND 8, 5
Break groups, TUT 12; HND 10
BREAK option, FOR EACH statement, TUT 12; REF
Broker, SA II Appendix C, 2
BTOS/CTOS concepts, SA I 4
BTOS/CTOS, customizing for PROGRESS, SA I 4
Buffers, TUT 8, 9; SA II 1,3,B,C
Building PROGRESS executables, 3GL 1
BY option, FOR EACH statement, TUT 4, 9; REF

C

CAN-DO function, HND 11; SA II 5; REF
CENTERED option
 DISPLAY statement, TUT 11; REF
 FORM statement, TUT 6; REF
Character constant, TUT 10
Character data type, TUT 5; HND 4
Character set, HND 2
CHOOSE statement, HND 7; REF
Client(s)
 local, SA II 2,3, Appendix C
 remote, SA II 2,3, Appendix C
C entry points, 3GL 13
COBOL entry points, 3GL 15
COLON option, UPDATE statement, TUT 11; REF
Color in screen displays, TUT 11
COLOR MESSAGES option, DISPLAY statement, TUT 11; REF
COLOR statement TUT 11; REF
Column. *See* Field.
Column labels, TUT 5, 9; HND 7
COLUMN-LABEL option, TUT 10

Command(s)

BTOS/CTOS, SA I 4

DOS, SA I 1

UNIX, SA I 2

VMS, SA I 3

for copying databases, SA II 2

for creating an empty database,
SA II 2

for deleting databases, SA II 2

for removing log file entries, SA II 4

for starting PROGRESS, TUT 1; SA II 2

procedure for starting an application, TUT 16

proutil, SA II 4, 7

Command line piping, for UNIX, SA I 1

Comments in procedures, TUT 4

Comparing character strings, TUT 9

COMPILE statement, TUT 13; HND 11; REF

Compiler limits, SA II 1

Compiling procedures, TUT 13;
HND 11

Compound statements, TUT 8

Concatenate operator (+), TUT 10; REF

Conditional processing, TUT 8

CONNECT statement, HND 13; REF; DG 2, 4

Connected database, TUT 3; HND 13; DG 2, 4

Connecting databases, HND 13; DG 2, 4

Connection overhead, HND 13

Connection parameters, HND 13

Constants, TUT 10

Continuation lines in the editor, TUT 2

Control break reports, TUT 12;
HND 10

Converting
databases, SA II 4,6
data, TUT 10

Copying
data definitions, HND 3; SA II 4
databases, SA II 2
demo database, TUT 4
files, HND 3

CREATE statement, TUT 8; REF

CREATE ALIAS statement, HND 13; REF

Creating

- databases, TUT 1; SA II 2, 6
- fields, TUT 5; HND 3
- files, TUT 5; HND 3
- indexes, TUT 5; HND 3

cron, SA I 1

Cross-tab reports, HND 10

D

Data definitions

- changing, TUT 3, 5; HND 3
- creating, TUT 5
- dumping and loading, SA II 4

Data Dictionary, TUT 1, 3, 5

- accessing, TUT 3
- accessing in Help procedure, TUT 15
- changing definitions, TUT 3; HND 3
- changing your password, HND 11
- defining files, fields and indexes, TUT 3, 5
- defining access permissions, HND 11; SA II 5; TK 5
- dumping and loading data, SA II 4
- for Query/Run-Time PROGRESS, TUT 5
- help line, TUT 3
- moving around in, TUT 3
- using the data exchange facility, SA II 4

Data movement, TUT 8; HND 1

Data types, TUT 5; HND 4

Database(s)

- active, TUT 3
- backing up, SA II 4
- concepts, TUT 1; HND 1
- creating, TUT 1; SA II 2, 6; TK 7
- disk usage, SA II 1, 4
- dumping and loading, SA II 4
- files, SA II 1
- limits, SA II 1
- multi-volume, SA II 6
- restoring, SA II 4, 7
- restricting access, TK 7
- security, HND 11; SA II 5; TK 3, 5
- status, TK 7
- stopping the server, SA II 2; SA I 6
- transporting, HND 14; TK 7

Database gateways, HND 13; DG 1, 2, 4

Database records, sharing, HND 12

Database types, HND 13; DG 2, 4

Data Definition Language, HND 15

Date arithmetic, TUT 10

Date data type, TUT 5; HND 4

DBRESTRICTIONS function, HND 13; REF; DG 2, 4

DBTYPE function, HND 13; REF

DBVERSION function, HND 13; REF; DG 2

Debugging applications, HND 14

Decimal data type, TUT 5; HND 2

DECnet networks, SA I 8

DEFINE BUFFER statement, TUT 10, REF

DEFINE SHARED FRAME statement, HND 4; REF

DEFINE STREAM statement, HND 9; REF

DEFINE VARIABLE statement, TUT 10, 14; REF

DEFINE WORKFILE statement, HND 10; REF

DELETE ALIAS statement, HND 13; REF

DELETE statement, TUT 4, 8; REF

Deleting

records, TUT 4

Demo database, TUT 1

default directory, TUT 13

definitions, TUT Appendix A

Designing

databases, TUT 5; HND 3; TK 7

screens, TUT 11; TK 5

Developer's Toolkit, TK

Dictionary. *See* Data Dictionary.

DICTIONARY statement, TUT 5; REF

DIF files, converting, HND 9

Directories

BTOS/CTOS, SA I 4

DOS, SA I 1

for application release, TK 8

UNIX, SA I 2

VMS, SA I 3

PROGRESS, SA II 1

DISCONNECT statement, HND 13; REF; DG 2, 4

Disk usage, SA II 1

Display formats, TUT 5; HND 4

DISPLAY statement, TUT 4, 11, 12

Distributed multi-database configuration, HND 13

Distributed/Simultaneous multi-database configuration, HND 13

Distributing applications, TUT 16;

HND 14; TK 3, 8

DLC directory, TUT 13
DO statement, TUT 4, 8, 9; REF
DOS concepts, SA I 1
 for using the Toolkit on DOS, TK 2
DOS, customizing for PROGRESS, SA I 2
DOS LANs, SA I 6, Appendix A
DOS memory saver (DMS), 3GI 1
DOS protected mode, SA I 2; 3GL 1
DOWN option, DISPLAY statement, TUT 11
Dumping a database, SA II 4
 for distribution, TK 7

E

EDITING phrase, HND 6; REF
Editor, TUT 2
 accessing, TUT 16
 area, TUT 1
Embedded SQL, HND 15; 3GL 9
Empty database, TUT 1; SA II 2
ENDKEY, HND 6, 8
END-ERROR key, TUT 16; HND 8
Environment variables
 BTOS/CTOS, SA I 4
 DOS, SA I 2
 OS/2, SA I 5
 UNIX, SA I 1
 VMS, SA I 3
ERROR key, HND 8
Error messages, TUT 4
Error processing, HND 8
Example procedures, TUT 14; HND Preface; REF Introduction
EXCLUSIVE-LOCK option, Record phrase, HND 12; REF
Exiting from PROGRESS, TUT 8
 from the editor, TUT 1
EXPORT statement, HND 9; REF
Expressions, TUT 10
Extended alphabet support, HND 2
Extents, SA II 6

F

Fields, TUT 1, 5, 11; HND 3

Federated multi-database configuration, HND 13

File(s)

- after-image, SA II 1, 7
- before-image, HND 8; SA II 1
- BTOS/CTOS, SA I 4
- creating, TUT 5; HND 3
- Data Dictionary definition, TUT 5; HND 3
- database, SA II 1
- DOS, SA I 2
- dumping and loading, SA II 4
- freezing, HND 11
- include, TUT 14
- limits, SA II 1
- lock control, SA II 1
- log, SA II 1
- maintenance, TUT 4
- moving data definitions between, HND 3; SA II 4
- multi-user, SA II 1
- OS/2, SA I 5
- permissions, HND 11; SA II 5
- relationships, TUT 1, 6
- UNIX, SA I 1
- version, TK 8
- VMS, SA I 3

FIND statement, TUT 4, 6, 8, 9; REF

FIRST option, FIND statement, TUT 9; HND 10; REF

Footers, TUT 12

FOR EACH statement, TUT 4, 6, 8, 9; REF

FOR option

- DO statement, TUT 9; REF

- REPEAT statement, TUT 8, 9; REF

FORM statement, TUT 6, 11, 12; HND 7; REF

FORMAT option

- DEFINE VARIABLE statement, TUT 10; REF

- DISPLAY statement, TUT 11; REF

- UPDATE statement, TUT 11; REF

Format phrase, TUT 11; HND 7; REF

Formats, display, TUT 5, 11; HND 4

FRAME option, REPEAT statement, TUT 11; REF

Frame phrase, TUT 11; HND 7; REF

Frames, TUT 11; HND 7

Function keys, TUT 2, 9; HND 6

Functions,

AMBIGUOUS, TUT 10; REF
AVAILABLE, TUT 10; REF
BEGINS, TUT 9; REF
ENTRY, TUT 10; REF
in expression, TUT 10
FRAME-COLUMN, TUT 11; REF
FRAME-DOWN, TUT 11; REF
FRAME-FILE, TUT 15; REF
FRAME-LINE, TUT 11; REF
FRAME-ROW, TUT 11; REF
FRAME-FIELD, TUT 15; REF
KEYCODE, TUT 9; REF
LAST-OF, TUT 12; REF
LASTKEY, TUT 9; REF
MATCHES, TUT 9; REF
OPSYS, TUT 10; REF
PAGE-NUMBER, TUT 12; REF
ROUND, TUT 10; REF
SQRT, TUT 10; REF
STRING, TUT 10; REF
SEARCH, TUT 15; REF
statistical, TUT 10
WEEKDAY, TUT 10; REF

G

GATEWAYS function, REF
GETBYTE, REF
Global variable(s)
 shared, TUT 14
GO-ON option, UPDATE statement, TUT 9

H

HEADER option, FORM statement, TUT 12; REF
Header(s)
 block, TUT 4
 frame, TUT 11
 in reports, TUT 12
Help
 for a field, TUT 5, 15; HND 3
 for Multi-db applications, HND 13
HELP option, SET statement, TUT 11; REF
High Level Call Interface (HLC), 3GL 1
 programming considerations, 3GL 4
 general programming, 3GL 3
HIDE statement, TUT 6, 12; HND 7
Hiding frames, TUT 11; HND 7

HLC libraries, 3GL 5

Host Language Interface (HLI), 3GL

C programming, 3GL 10

COBOL programming, 3GL 12

Pascal programming, 3GL 14

I

IF...THEN...ELSE statement, TUT 6, 8; REF

IMPORT statement, HND 9; REF

Improving performance, SA II 3,B

Include files, TUT 14

Indexes, TUT 1

defining, TUT 5; HND 3

using for record selection, TUT 9

using for sorting records, TUT 9

Input

redirecting, TUT 15

sources, HND 9

INPUT FROM statement, TUT 15

INSERT statement, TUT 4, 8; REF

Integer data type, TUT 5

Integrity, TUT 5

K

Key(s)

codes, HND 2

defining, HND 2, 6

functions, HND 6

labels, HND 2

PROGRESS editor, TUT 2

using in procedures, HND 6

KEYCODE function, HND 2, 6; REF

KEYFUNCTION function, HND 2, 6; REF

KEYLABEL function, HND 2, 6; REF

Keystrokes, monitoring, HND 6

L

Labels

for columns, TUT 5, 11

for keys, HND 2

LAST option, FIND statement, TUT 9; REF

LASTKEY function, HND 6; REF

LDBNAME function, HND 13; REF
LENGTH function, REF
LENGTH statement, REF
LIBRARY function, REF
Libraries, SA II 4
Limits, SA II 1
Loading data, SA II 4
Local-area networks (LANs), SA I 5,6,Appendix A
Local-before-image file (.lbi), HND 8; SA II 1
Log file (.lg), SA II 1,4
Logical database, HND 13
Logical data type, TUT 5; HND 4
Logicals, VMS. *See* Environment variables.
Login procedures, HND 11; TK 5
Locks, record, HND 12
LOOKUP function, HND 6; REF
Looping, TUT 4; HND 5

M

MATCHES function, TUT 9; REF
MEMBER function, REF
Memory, shared. *See* Shared Memory.
Memory usage, SA II 3
Menus, TUT 6; HND 7
MESSAGE statement, TUT 6, 11; REF
Messages, TUT 4, 11; HND 1
Monitor, SA II Appendix B
Multi-database configuration(s)
 federated, HND 13
 distributed, HND 13
 distributed/simultaneous, HND 13
Multi-database programming, HND 13
Multiple application processors, SA I 1
Multi-threaded architecture, SA II Appendix C
Multi-user PROGRESS, HND 12
Multi-user client connection mode, HND 13
Multi-user direct access mode, HND 13

N

Named Pipes, HND Appendix A

NETBIOS, SA I 6

Networks, SA II 1; SA I 5-11

NEXT option, FIND statement, TUT 9; REF

NEXT statement, TUT 8; HND 6; REF

NO-ATTR-SPACE option, Frame phrase, HND 7; REF; TK 5

NO-BOX option, Frame phrase, TUT 11; HND 7; REF

NO-ECHO option, INPUT FROM statement, TUT 15; REF

NO-ERROR option, FIND statement, TUT 8; REF

NO-HIDE option, Frame phrase, HND 7; REF

NO-LOCK option, Frame phrase, HND 12

Non-PROGRESS databases, HND 13

Normalizing data, HND 3

Null string, TUT 10, 15

Numeric calculations, TUT 10

O

Object (.r) file, TUT 13

OF option, FOR EACH statement, TUT 6, 9; REF

ON statement, HND 6, 8; REF

Operating system(s)

- administration, SA II all

- BTOS/CTOS, SA I 4

- differences between, HND 14

- DOS, SA I 1

- facilities to backup and restore databases, SA II 4

- OS/2, SA I 5

- security, HND 11; SA II 5

- transporting databases, TK

- UNIX, SA I 2

- VMS, SA I 3

- writing applications for different, HND 14

Operators, TUT 10

OPSYS function, TUT 10; REF

OS/2, SA I 5

OUTPUT CLOSE statement, TUT 6; HND 9; REF

Output destinations, HND 9

OUTPUT THROUGH statement, HND 9; REF
OUTPUT TO statement, TUT 6, 12; HND 9; REF
Overlay frame, TUT 11
OVERLAY option, Frame phrase, TUT 11; HND 7

P

PAGE-BOTTOM option, FORM statement, TUT 12; REF
PAGE-TOP option, FORM statement, TUT 12; REF
PAGED option, OUTPUT statement, HND 9; REF
Parameter files, SA II 3; HND 13
Pascal entry points, 3GL 15
Passwords, HND 11; SA II 5; TK 5
PDBNAME function, HND 13; REF
Performance consideration, UNIX, SA I 1
Permissions, HND 11; SA II 5; TK 3, 5; SA I 1, 3
Pipes, named, HND Appendix A
PREV option, FIND statement, TUT 9; REF
Printing, TUT 12; HND 9
Privileges, HND 11; SA II 5
PROBUILD utility, 3GL 1
PROGRESS executable, building, 3GL 1
Procedures
 applhelp.p, HND 3
 compiling, TUT 13
 creating, TUT 4
 debugging, HND 14
 gateway
 precompiling, TUT 13; TK 7
 running, TUT 4
 sample, TUT 4; HND Preface; REF Introduction
 time stamp, TUT 13
 using keys in, HND 6
PROGRESS
 database commands, TUT 1; SA II 1, 2
 editor, TUT 2
 keys, TUT 2, HND 2
 leaving, TUT 1
 limits, SA II 1
 multi-user, HND 12
 multi-volume, SA II 6
 starting, TUT 1; HND 12; SA II 2; SA I
 startup options, SA II 3
 utilities, SA II

Prolib utility, SA II 4

PROMPT-FOR statement, TUT 4, 8; REF

PROPATH environment variable,
TUT 13, 14; SA I 1, 2,3,4

protected mode, for DOS, SA I 2; 3GL 1

protermcap file, TUT 4; HND 2; TK 3

Programming tips, HND 14

PUTBYTE function, REF

PUT statement, TUT 12; REF

Q

Query/Run-Time PROGRESS, Data Dictionary, TUT 5

QUIT statement, TUT 8, 13; REF

Quoter, HND 9

R

RAW function, REF

RAW statement, REF

Raw Datatype, HND 13, DG 2

READKEY statement, HND 6; REF

Rebuilding indexes,

Record(s)

- buffer, TUT 8, 9

- defining, HND 3

- limits, SA II 1

- locks, HND 8, 12

- phrase, HND 12; REF

- retrieving, TUT 4, 9

- scope, TUT 9; HND 5

Recovery, database, SA II 4, 7

Reloading databases, SA II 4

Remote database, HND 13

REPEAT statement, TUT 4, 8, 9

Reports, TUT 12; HND 10

Resource usage, SA II 3,B,C

Restoring a database, SA II 4, 7

RETAIN option, DISPLAY statement, TUT 11; HND 7; REF

Roll Forward Recovery, SA II 7

ROUND function, TUT 10; REF

Row. *See* Record.

ROW option, DISPLAY statement, TUT 11; REF

RUN statement, TUT 6, 14; REF

S

Sample procedures, TUT 4;

HND Preface; REF Preface

SAVE option, COMPILE statement, TUT 13; REF

Schema, SA II 1; TK 4; DG 1, 2, 4; *See also* Data Dictionary

Schema holder database, HND 13; DG 1, 2, 4

Scoping

for frames, HND 5, 9

for records, TUT 8; HND 5

Screen

attributes, TUT 11; TK 5

buffer, TUT 8

SCROLL option, Frame phrase, HND 7; REF

SCROLL statement, HND 7; REF

SDBNAME function, HND 13; REF

SEARCH function, TUT 15; HND 9; REF

Search paths, VMS, SA I 3

Security, TUT 5; HND 11; SA II 5; TK 3, 5; SA I 1, 2, 3, 4

Servers HND 12; SA II 2; SA I 1, 3, 4, 7, 8

SET statement, TUT 8, 11; REF

SETUSERID function, HND 11; REF

SHARE-LOCK option, Record phrase, HND 12; REF

Shared frames, HND 7

Shared memory, SA II Appendix C, Appendix B, SA I 1, 3

Shared variables, TUT 14, 15

Shutting down a database, SA II 2

SIDE-LABELS option, UPDATE statement, TUT 6; REF

Single-user connection mode, HND 13

SKIP option,

DISPLAY statement, TUT 11; REF

FORM statement, TUT 6; REF

Sorting records, TUT 9; HND 10

SPACE option, UPDATE statement, TUT 11; REF

SPX, SA I 6

SQRT function, TUT 10; REF

STATUS statement, TUT 11; REF
Starting PROGRESS, TUT 1; SA II 2; SA I
Startup options, SA II 3
Stopping PROGRESS, SA II 2
Streams, HND 9
STRING function, TUT 10, 12; REF
Structure file, SA II 6
Subprocedures, TUT 14
SUBSTRING function, HND 9; REF
Subtransactions, HND 8
SYLK files, converting, HND 9
System administration, SA II; SA I
System failure, HND 8; SA II 4,7

T

Table. *See* File.
TCP/IP networks, SA I 7
Terminals, definition of, HND 2; SA I 1, 3, 4
Terminal support, TUT 11; HND 7, 13; TK 3, 5
Testing
 applications, HND 14
 backups, SA II 4
TEXT option, UPDATE statement, TUT 11; REF
Time display formats, HND 4
Time stamp, TUT 13
TITLE option
 DISPLAY statement, TUT 11; REF
 FORM statement, TUT 6; REF
TO option, UPDATE statement, TUT 11; REF
TOP-ONLY option, Frame phrase, HND 7; REF
TOTAL BY option, DISPLAY statement, TUT 12; REF
Totals, in reports, TUT 12; HND 10
Transactions, HND 8, 12
Transporting applications, HND 14; TK 3, 8
Troubleshooting, SA II Appendix A
Two-way streams, UNIX, SA I 1

U

Undo processing, HND 5, 8

UNDO statement, HND 8; REF

UNIX concepts, SA I 1
for using the Toolkit on UNIX, TK 2

UNIX, customizing for PROGRESS, SA I 1

Unknown value, TUT 5

Unpacking sample procedures, TUT 4; HND Preface; REF Introduction

UPDATE statement, TUT 4, 6, 11; REF

Upgrading applications, TK 9

User-Defined language rules, HND 2

Utilities
database, SA II 4, 7
Developer's Toolkit, TK 1, 11

_User file, HND 11; SA II 5; TK 5

Userids, HND 11; SA II 5; TK 5

USE-INDEX option
FIND statement, TUT 9; REF
FOR EACH statement, TUT 9; REF

USING option, FIND statement, TUT 9; REF

V

VALIDATE option, SET statement, TUT 11; REF

Validation, TUT 5; HND 3

Variables, environment. *See* Environment Variables.

Variables, TUT 10, 11, 14

Views, HND 15

VIEW statement, TUT 11, 12; REF

VMS concepts, SA I 3
for using the Toolkit on VMS, TK 2

VMS, customizing for PROGRESS, SA I 3

W

WEEKDAY function, TUT 10; REF

WHERE option, FOR EACH statement, TUT 4, 6, 9; REF

Wildcard characters
BTOS/CTOS, SA I 4
DOS, SA I 1
UNIX, SA I 2
VMS, SA I 3































WITH option, Frame phrase, TUT 11; REF

Work files, HND 10

X

X Windows, HND 16

PROGRESS FUNCTION AND CONTROL KEYS

| | | | | | | | | | | | | | | | | |
|-----------------------|---|---|---|---|---|---|--|---|---|---|---|---|---|---|---|---|
| EDIT KEYS | GO | HELP | INSERT MODE | RIGHT-END | GET | PUT | RECALL | CLEAR | NEW LINE | DELETE LINE | BREAK LINE | APPEND LINE | FIND | BLOCK | PAGE-UP | PAGE-DOWN |
| EXECUTION KEYS | GO | HELP | INSERT MODE | END-ERROR | | | RECALL | CLEAR | | | | | | | | |
| Function Keys |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Control Keys |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | |

PROGRESS END: Type quit

| | |
|------------|--------------|
| DOS | |
| DOS END: | Type exit |
| STOP: | CTRL-BREAK |
| ABORT: | CTRL-ALT-DEL |

| | | |
|-----------|-------------|---------------------------------|
| | UNIX | |
| UNIX END: | CTRL-D | } Depends on UNIX stty settings |
| STOP: | CTRL-C | |
| ABORT: | CTRL-\ | |

| | |
|------------|--------------|
| VMS | |
| VMS END: | Logout |
| STOP: | CTRL-C |
| ABORT: | CTRL-Y, STOP |

| | |
|-------------|-------------------|
| BTOS | |
| BTOS END: | Run PROGRESS EXIT |
| STOP: | ACTION-CANCEL |
| ABORT: | ACTION/ |