INFORMIX-4GL

C Compiler Version
Rapid Development System
Interactive Debugger

Quick Reference

Release 4.1

Part No. 000-7024

Information in this **INFORMIX-4GL** *Quick Reference Card is applicable to all* **INFORMIX-4GL** *products, including* **INFORMIX-4GL Rapid Development System**, **INFORMIX-4GL Interactive Debugger**, *and* **INFORMIX-4GL (C Compiler Version)**.

The **INFORMIX-4GL Interactive Debugger** *is compatible with only the* **INFORMIX-4GL Rapid Development System**.

The information in this publication was compiled as of June 1991 and is subject to change without notice.

INFORMIX-4GL

Conventions

UPPERCASE LETTERS	denote keywords, which you must enter as shown. (You can use lowercase letters.)
italics	denote terms for which you must substitute identifiers or expressions.
[ ] and { }	denote options. Vertical bars separate the options. Braces ({ }) mean you *must* choose one option. Brackets ([ ]) mean you can, but are not required to, choose one option.
|	denotes a choice among several options.
. . .	denotes optional repetition of the previous clause.
UNDERLINE	denotes the default option.

Data Types

ARRAY [*i, j, k*] OF *type*
OL BYTE
CHAR (*n*)
CHARACTER (*n*)
DATE
DATETIME *first* TO *last*
DEC [(*m* [,*n*])]
DECIMAL [(*m* [,*n*])]
DOUBLE PRECISION [(*n*)]
FLOAT [(*n*)]
INT
INTEGER
INTERVAL *f* [(*p*)] TO *l*
LIKE *table.column*
MONEY [(*m* [,*n*])]
NUMERIC [(*m* [,*n*])]
REAL
RECORD {LIKE *table*.*
| *variable-list datatype*
[, ...] END RECORD}
SERIAL [(*n*)]
SMALLFLOAT
SMALLINT
OL TEXT
OL VARCHAR [(*m* [,*n*])]

OL This icon indicates data types specific to **INFORMIX-OnLine**.

Operations on Variables

Number Operations

+	addition
–	subtraction
*	multiplication
/	division
**	exponentiation
MOD	modulus
( )	associative
USING	formatting

String Operations

,	concatenation
[*m*,*n*]	substring
CLIPPED	drop trailing blanks
WORDWRAP	multiple line display

Relational

=	equal to
!= or <>	not equal to
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to

Boolean Expressions

expr rel-op expr

char-expr [NOT] LIKE *char-expr*
[ESCAPE "*escape-character*"]

char-expr [NOT] MATCHES *char-expr*
[ESCAPE "*escape-character*"]

expr IS [NOT] NULL

[NOT] *Boolean-expr*

Boolean-expr {AND | OR} *Boolean-expr*

Global Variables and Constants

TRUE
FALSE
INT_FLAG
QUIT_FLAG
STATUS

```
SQLCA RECORD
  SQLCODE    INT,
  SQLERRM    CHAR(71),
  SQLERRP    CHAR(8),
  SQLERRD    ARRAY[6] OF INT,
  SQLAWARN   CHAR(8)
END RECORD
```

Built-in Functions

ASCII *integer-expr*
char-expr CLIPPED
COLUMN *integer-expr*
CURRENT [*first* TO *last*]
DATE
DATE (*dtime-expr*)
DAY (*dtime-expr*)
EXTEND (*dtime-expr* [, *first* TO *last*])
LENGTH (*char-expr*)
MDY (*expr*, *expr*, *expr*)
MONTH (*dtime-expr*)
TIME
TIME (*dtime-expr*)
TODAY
integer-expr UNITS *qualifier*
USER
expr USING "*format-string*"
WEEKDAY (*dtime-expr*)
YEAR (*dtime-expr*)

Library Functions

ARG_VAL (*integer-expr*)
ARR_COUNT ( )
ARR_CURR ( )
DOWNSHIFT (*char-expr*)
ERR_GET (*integer-expr*)
ERR_PRINT (*integer-expr*)
ERR_QUIT (*integer-expr*)
ERRORLOG (*char-expr*)
FGL_GETENV (*char-expr*)
FGL_KEYVAL (*char-expr*)
FGL_LASTKEY ( )
FIELD_TOUCHED ({*field-list* | *screen-record*.*} [,...])
GET_FLDBUF ({*field-list* | *screen-record*.*} [,...]
INFIELD (*field-name*)
LENGTH (*char-expr*)
NUM_ARGS ( )
SCR_LINE ( )
SET_COUNT (*integer-expr*)
SHOWHELP (*integer-expr*)
STARTLOG ("*filename*")
UPSHIFT (*char-expr*)

Privileges

Database Level

CONNECT	DBA	RESOURCE

Table Level

ALTER	INDEX	SELECT [(*column-name*, ...)]
DELETE	INSERT	UPDATE [(*column-name*, ...)]

Display Attributes

WHITE	UNDERLINE
YELLOW	NORMAL
MAGENTA	BOLD
RED	REVERSE
CYAN	BLINK
GREEN	DIM
BLUE	INVISIBLE
BLACK	LEFT

4GL Statement Syntax

Following are statements for use with both **INFORMIX-SE** and **INFORMIX-OnLine**.

SE This icon indicates statements specific to **INFORMIX-SE**.

OL This icon indicates data types specific to **INFORMIX-OnLine**.

```
ALTER INDEX index-name TO [NOT] CLUSTER

SE ALTER TABLE table-name
    {
        ADD
            (newcol-name newcol-type [NOT NULL]
                [UNIQUE [CONSTRAINT constr-name]]
                [, ...])
            [BEFORE oldcol-name]
        |
        DROP (oldcol-name [, ...])
        |
        MODIFY (oldcol-name newcol-type
            [NOT NULL] [, ...])
        |
        ADD CONSTRAINT UNIQUE
            (oldcol-name [, ...])
            [CONSTRAINT constr-name]
        |
        DROP CONSTRAINT (constr-name [, ...])
    } [, ...]
```

```
OL ALTER TABLE table-name
     {
        ADD (newcol-name
           {
              newcol-type
              |
              {BYTE | TEXT}
                 [IN {TABLE | blobspace-name}]
           }
           [NOT NULL]
           [UNIQUE [CONSTRAINT constr-name]]
           [, ...])
           [BEFORE oldcol-name]
        |
        DROP (oldcol-name [, ...])
        |
        MODIFY (oldcol-name newcol-type
           [NOT NULL] [, ...])
        |
        ADD CONSTRAINT UNIQUE
           (oldcol-name [, ...])
           [CONSTRAINT constr-name]
        |
        DROP CONSTRAINT (constr-name [, ...])
        |
        LOCK MODE ({PAGE | ROW})
        |
        MODIFY NEXT SIZE number
     }[, ...]
  BEGIN WORK
  CALL function ([argument-list])
     [RETURNING variable-list]
  CASE [expr]
     WHEN {expr | Boolean-expr}
        {statement | EXIT CASE} ...
     ...
     [OTHERWISE {statement | EXIT CASE} ...]
     END CASE
  CLEAR
     {
        SCREEN
        |
        WINDOW window-name
        |
        FORM
        |
        field-list
     }
  CLOSE cursor-name
```

```
CLOSE DATABASE
CLOSE FORM form-name
CLOSE WINDOW window-name
COMMIT WORK
CONSTRUCT
   {
      BY NAME char-variable ON column-list
      |
      char-variable ON column-list FROM
         {field-list | screen-record [[n]].*} [, ...]
   }
   [ATTRIBUTE (attribute-list)]
   [HELP help-number]
   [
      {
         {
            BEFORE CONSTRUCT
            |
            AFTER CONSTRUCT
            |
            BEFORE FIELD field-list
            |
            AFTER FIELD field-list
            |
            ON KEY (key-list)
         }
         {
            statement
            |
            NEXT FIELD
               {
                  field-name
                  |
                  NEXT
                  |
                  PREVIOUS
               }
            |
            CONTINUE CONSTRUCT
            |
            EXIT CONSTRUCT
         } ...
      } ...
      END CONSTRUCT
   ]
```

```
CONTINUE {FOR | FOREACH | MENU | WHILE}
```

SE
```
CREATE AUDIT FOR table-name IN "pathname"
```

SE
```
CREATE DATABASE database-name
    [WITH LOG IN "pathname" [MODE ANSI]]
```

OL
```
CREATE DATABASE database-name
    [IN dbspace-name]
    [
        WITH
        {
            [BUFFERED] LOG
            |
            LOG MODE ANSI
        }
    ]
```

```
CREATE [UNIQUE] [CLUSTER]
    INDEX index-name ON table-name
        (column-name [ASC | DESC] [, ...])
```

```
CREATE SYNONYM synonym FOR table-name
```

SE
```
CREATE [TEMP] TABLE table-name
    (column-name datatype
    [NOT NULL]
    [
        UNIQUE [(unique-col-list)]
            [CONSTRAINT constr-name]
    ] [, ...])
    [WITH NO LOG]
    [IN "pathname"]
```

OL
```
CREATE [TEMP] TABLE table-name
    (column-name
        {
            datatype
            |
            {BYTE | TEXT}
                [IN {TABLE | blobspace-name}]
        }
        [NOT NULL]
        [
            UNIQUE [(unique-col-list)]
                [CONSTRAINT constr-name]
        ] [, ...])
    [WITH NO LOG]
    [IN dbspace-name]
    [EXTENT SIZE extent-size]
    [NEXT SIZE next-size]
    [LOCK MODE ({PAGE | ROW})]
```

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

CURRENT WINDOW IS {window-name | SCREEN}

DATABASE database-name [EXCLUSIVE]

DECLARE cursor-name
   {
      CURSOR [WITH HOLD] FOR
         {
            SELECT-statement
               [FOR UPDATE [OF column-list]]
            |
            INSERT-statement | statement-id
         }
      |
      SCROLL CURSOR [WITH HOLD] FOR
         {SELECT-statement | statement-id}
   }

DEFER {INTERRUPT | QUIT}

DEFINE variable-list
   {
      datatype
      |
      LIKE table.column
      |
      RECORD
         {
            LIKE table.*
            |
            variable-list datatype [, ...] END RECORD
         }
   } [, ...]

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

```
DISPLAY
   {
      BY NAME variable-list
      |
      variable-list
         [
            TO {field-list | screen-record [[n]].*} [, ...]
            |
             AT screen-row, screen-column
         ]
   }
   [ATTRIBUTE (attribute-list)]

DISPLAY ARRAY record-array TO screen-array.*
   [ATTRIBUTE (attribute-list)]
   {
      ON KEY (key-list)
         {
            statement
            |
            EXIT DISPLAY
            |
            END DISPLAY
         } ...
      |
      [END DISPLAY]
   }

DISPLAY FORM form-name
   [ATTRIBUTE (attribute-list)]

SE DROP AUDIT FOR table-name

DROP DATABASE {database-name | char-variable}

DROP INDEX index-name

DROP SYNONYM synonym

DROP TABLE table-name

DROP VIEW view-name

ERROR display-list [ATTRIBUTE (attribute-list)]

EXECUTE statement-id [USING input-list]
```

```
EXIT
   {
      CASE
      |
      DISPLAY
      |
      FOR
      |
      FOREACH
      |
      INPUT
      |
      MENU
      |
      PROGRAM [(integer-expr)]
      |
      WHILE
   }
FETCH
   [
      NEXT
      |
      {PREVIOUS | PRIOR}
      |
      FIRST
      |
      LAST
      |
      CURRENT
      |
      RELATIVE integer
      |
      ABSOLUTE integer
   ]
   cursor-name [INTO variable-list]
FINISH REPORT report-name
FLUSH cursor-name
```

```
FOR integer-var = integer-expr TO integer-expr
   [STEP integer-expr]
   {
      statement
      |
      CONTINUE FOR
      |
      EXIT FOR
   } ...
   END FOR

FOREACH cursor-name [INTO variable-list]
   {
      statement
      |
      CONTINUE FOREACH
      |
      EXIT FOREACH
   } ...
   END FOREACH
```

SE `FREE {statement-id | cursor-name}`

OL `FREE {statement-id | cursor-name | blob-variable}`

```
FUNCTION function-name ([argument-list])
   {statement |RETURN expr-list} ...
   END FUNCTION

GLOBALS
   {
      "filename"
      |
      DEFINE-statement
         ...
         END GLOBALS
   }

GOTO [ : ] label-id

GRANT table-privilege ON table-name
   TO {PUBLIC | user-list}
   [WITH GRANT OPTION]
   [AS user]

GRANT database-privilege TO {PUBLIC | user-list}

IF Boolean-expr THEN
   statement ...
   [ELSE statement ...]
   END IF

INITIALIZE variable-list
   {LIKE column-list | TO NULL}
```

```
INPUT
   {
      BY NAME variable-list
         [WITHOUT DEFAULTS]
      |
      variable-list
         [WITHOUT DEFAULTS]
         FROM
         {field-list | screen-record [[n]].*} [, ...]
   }
   [ATTRIBUTE (attribute-list)]
   [HELP help-number]
   [
      {
         {
            BEFORE INPUT
            |
            AFTER INPUT
            |
            BEFORE FIELD field-list
            |
            AFTER FIELD field-list
            |
            ON KEY (key-list)
         }
         {
            statement
            |
            NEXT FIELD
               {
                  field-name
                  |
                  NEXT
                  |
                  PREVIOUS
               }
            |
            CONTINUE INPUT
            |
            EXIT INPUT
         } ...
      } ...
      END INPUT
   ]
```

```
INPUT ARRAY record-array
   [WITHOUT DEFAULTS] FROM screen-array.*
   [HELP help-number]
   [ATTRIBUTE (attribute-list)]
   [
      {
         {
            BEFORE
               {INPUT | ROW | INSERT |DELETE}
            |
            AFTER
               {INPUT | ROW | INSERT |DELETE}
            |
            BEFORE FIELD field-list
            |
            AFTER FIELD field-list
            |
            ON KEY (key-list)
         }
         {
            statement
            |
            NEXT FIELD
               {field-name | NEXT |PREVIOUS}
            |
            CONTINUE INPUT
            |
            EXIT INPUT
         } ...
      } ...
      END INPUT
   ]
INSERT INTO table-name [(column-list)]
   {VALUES (value-list) | SELECT-statement}
LABEL label-id:
LET variable = expr
LOAD FROM "pathname" [DELIMITER "char"]
   {
      INSERT INTO table-name
         [(column-name [, ...])]
      |
      INSERT-statement
   }
```

```
OL LOCATE variable-list IN
     {
        MEMORY
        |
        FILE [filename]
     }
   LOCK TABLE table-name
     IN {SHARE | EXCLUSIVE} MODE
   MAIN
     statement
     ...
     END MAIN
   MENU menu-name
     {
        {
           BEFORE MENU
           |
           COMMAND
              {
                 KEY (key-list)
                 |
                 [KEY (key-list)] menu-option
                    [option-description]
                    [HELP help-number]
              }
        }
        {
        statement
        |
        CONTINUE MENU
        |
        EXIT MENU
        |
        NEXT OPTION menu-option
        |
        SHOW OPTION {option-list | ALL}
        |
        HIDE OPTION {option-list | ALL}
        } ...
     } ...
     END MENU
   MESSAGE display-list
     [ATTRIBUTE (attribute-list)]
```

```
OPEN cursor-name [USING variable-list]
OPEN FORM form-name FROM form-file
OPEN WINDOW window-name
   AT screen-row, screen-column
   WITH
   {
      integer ROWS, integer COLUMNS
      |
      FORM form-file
   }
   [ATTRIBUTE (attribute-list)]
OPTIONS
   {
      MESSAGE LINE line-value
      |
      PROMPT LINE line-value
      |
      MENU LINE line-value
      |
      COMMENT LINE line-value
      |
      ERROR LINE line-value
      |
      FORM LINE line-value
      |
      INPUT {WRAP | NO WRAP}
      |
      INSERT KEY key-name
      |
      DELETE KEY key-name
      |
      NEXT KEY key-name
      |
      PREVIOUS KEY key-name
      |
      ACCEPT KEY key-name
      |
      HELP FILE help-file
      |
      HELP KEY key-name
      |
      INPUT ATTRIBUTE (attribute-list)
      |
      DISPLAY ATTRIBUTE (attribute-list)
      |
      SQL INTERRUPT {ON | OFF}
      |
```

```
        FIELD ORDER
          {
            CONSTRAINED
            |
            UNCONSTRAINED
          }
      }
      [, ...]
  OUTPUT TO REPORT report-name (expr-list)
  PREPARE statement-id FROM string-spec
  PROMPT display-list
      [ATTRIBUTE (attribute-list)]
      FOR [CHAR] variable
      [HELP help-number]
      [ATTRIBUTE (attribute-list)]
      [
        ON KEY (key-list)
          statement
          ...
        ...
        END PROMPT
      ]
  PUT cursor-name [FROM variable-list]
SE RECOVER TABLE table-name
  RENAME COLUMN table.oldcol-name
      TO newcol-name
  RENAME TABLE oldname TO newname
  REPORT report-name (variable-list)
      < See REPORT Routines on page 23>
      END REPORT
  RETURN [expr-list]
  REVOKE
      {
        table-privilege ON table-name
        |
        database-privilege
      }
      FROM {PUBLIC | user-list}
  ROLLBACK WORK
SE ROLLFORWARD DATABASE database-name
```

RUN *command-line*
[
RETURNING *integer-variable*
|
WITHOUT WAITING
]

SCROLL {*field-list* | *screen-record*.*} [, ...]
{UP | DOWN}
[BY *integer*]

SELECT <See SELECT Statement on page 21>

SET EXPLAIN {ON | OFF}

OL SET ISOLATION TO
{
CURSOR STABILITY
|
{DIRTY | COMMITTED | REPEATABLE}
READ
}

SE SET LOCK MODE TO [NOT] WAIT

OL SET LOCK MODE TO
{NOT WAIT | WAIT [*seconds*]}

OL SET [BUFFERED] LOG

SLEEP *integer-expression*

SE START DATABASE *database-name*
WITH LOG IN "*pathname*"
[MODE ANSI]

START REPORT *report-name*
[TO {*filename* | PIPE *program* | PRINTER}]

UNLOAD TO "*pathname*"
[DELIMITER "*char*"]
SELECT-statement

UNLOCK TABLE *table-name*

```
UPDATE table-name SET
   {
      column-name = expr [, ...]
      |
      {(column-list) | [table-name.] *} =
         {(expr-list) | record-name.*}
   }
   [
      WHERE
         {
            condition
            |
            CURRENT OF cursor-name
         }
   ]
UPDATE STATISTICS [FOR TABLE table-name]
VALIDATE variable-list LIKE column-list
WHENEVER
   {
      [[ANY] ERROR | SQLERROR]
      |
      [WARNING | SQLWARNING]
      |
      NOT FOUND
   }
   {
      [GOTO | GO TO] [:] label
      |
      CALL function-name
      |
      CONTINUE
      |
      STOP
   }
WHILE Boolean-expr
   {statement |EXIT WHILE |CONTINUE WHILE}
   ...
   END WHILE
```

SELECT Statement

SELECT [ALL | [DISTINCT | UNIQUE]] *select-list*
  [INTO *variable-list*]
  FROM
  {
    table-name [*table-alias*]
    |
    OUTER *table-name* [*table-alias*]
    |
    OUTER (*table-expr*)
  } [, ...]
  [WHERE *condition*]
  [GROUP BY *column-list*]
  [HAVING *condition*]
  [ORDER BY *column-name* [ASC | DESC][, ...]]
  [INTO TEMP *table-name*]
  [WITH NO LOG]

SELECT-statement UNION [ALL]
  SELECT-statement
  [UNION [ALL] *SELECT-statement*] ...

Conditions

expr rel-op expr

expr [NOT] BETWEEN *expr* AND *expr*

expr [NOT] IN ({*value-list* | *SELECT-statement*})

column-name [NOT] LIKE "*string*"
  [ESCAPE "*escape-character*"]

column-name [NOT] MATCHES "*string*"
  [ESCAPE "*escape-character*"]

expr rel-op {ALL | [ANY | SOME]}
  (*SELECT-statement*)

[NOT] EXISTS (*SELECT-statement*)

column-name IS [NOT] NULL

[NOT] *condition*

condition {AND | OR} *condition*

Aggregate Functions

AVG({[[DISTINCT | UNIQUE] | ALL]
column-name | [ALL] *expr*})

COUNT({[DISTINCT | UNIQUE]
column-name | *})

MAX({[[DISTINCT | UNIQUE] | ALL]
column-name | [ALL] *expr*})

MIN({[[DISTINCT | UNIQUE] | ALL]
column-name | [ALL] *expr*})

SUM({[[DISTINCT | UNIQUE] | ALL]
column-name | [ALL] *expr*})

REPORT Routines

```
REPORT report-name (argument-list)
   [DEFINE-statement]
   [
      OUTPUT
         [
            REPORT TO
               {
                  "filename"
                  |
                  PIPE "program"
                  |
                  PRINTER
               }
         ]
         [LEFT MARGIN integer]
         [RIGHT MARGIN integer]
         [TOP MARGIN integer]
         [BOTTOM MARGIN integer]
         [PAGE LENGTH integer]
         [TOP OF PAGE "char-string"]
   ]
   [ORDER [EXTERNAL] BY variable [, ...]]
   FORMAT
      {
         EVERY ROW
         |
         {
            {
               [FIRST] PAGE HEADER
               |
               PAGE TRAILER
               |
               ON {EVERY ROW | LAST ROW}
               |
               {BEFORE | AFTER} GROUP OF variable
            }
            statement ...
         } ...
      }
   END REPORT
```

Report-Only Statements

NEED *integer-expr* LINES

PAUSE ["*string*"]

PRINT [[*expr-list*] [;] | FILE "*filename*"]

SKIP {*integer-expr* LINE[S] | TO TOP OF PAGE}

Report-Only Functions

[GROUP]
  {
    COUNT(*)
    |
    PERCENT(*)
    |
    {SUM | AVG | MIN | MAX} (*expression*)
  }
  [WHERE *Boolean-expr*]

LINENO

PAGENO

integer-expr SPACE[S]

char-expr WORDWRAP
  [RIGHT MARGIN *integer-expr*]

Form Specification Summary

Form Specification

Note: The braces and brackets around the *field-tag* are required in the SCREEN section.

DATABASE {*database* | FORMONLY}
  [WITHOUT NULL INPUT]

SCREEN [SIZE *lines* [BY *cols*]]
  {
    [*text*] [*field-tag*] [*graphics-char*]
    ...
  }
  [END]

```
[
   TABLES
      {
         table
         |
         table-alias = [database [@server] :]
            [owner.] table
      } ...
      [END]
]
ATTRIBUTES
   field-tag = field-description;
   ...
   [END]
[
   INSTRUCTIONS
      [DELIMITERS "ab"]
      [
         SCREEN RECORD record-name [[n]]
            (
               {
                  table-name.*
                  |
                  table-name.column1
                     THRU table-name.column2
                  |
                  table-name.column
               } [, ...]
            )
      ] ...
      [END]
]
```

Field Description

```
field-tag =
   {
      [table.]column
      |
      FORMONLY. field-name
         [
            TYPE
               [
                  datatype [NOT NULL]
                  |
                  LIKE [table.] column
               ]
         ]
   }
   [, attribute [, ...]] [= ...] [;] [= ...];
```

Attributes

AUTONEXT
COLOR = *disp-mode* ...
  [WHERE *Boolean-expr*]
COMMENTS = "*string*"
DEFAULT = *value*
DISPLAY LIKE
  table.column
DOWNSHIFT
FORMAT="*format-string*"
INCLUDE = (*value-list*)
INVISIBLE
NOENTRY
PICTURE =
  "*format-string*"
OL PROGRAM= "*name*"
REQUIRED
REVERSE
UPSHIFT
VALIDATE LIKE
  table.column
VERIFY
WORDWRAP
  [COMPRESS]

Map of Debugger Windows and Screens

Keys for Cursor Movement

From either Debugger screen window:

CONTROL-J	moves the cursor down one line
CONTROL-K	moves the cursor up one line
CONTROL-B	moves the cursor up one window
CONTROL-F	moves the cursor down one window
CONTROL-U	moves the cursor up one-half window
CONTROL-D	moves the cursor down one-half window
n[**CONTROL-***key*]	repeats cursor movement *n* times

From the Source window:

n[**RETURN**]	moves to line *n* of the source module
$	moves to last line of the source module
RETURN	moves to the next instance of the most recent search pattern

Command Syntax

Note: Except for literal braces around command strings in the syntax of ALIAS, BREAK, and TRACE, Debugger syntax conventions are as described for **INFORMIX-4GL**.

ALIAS
{
*
|
name = *cmd_str*
|
name = {*cmd_str* [; *cmd_str* ...]}
}

APPLICATION [DEVICE] *device-name*

BREAK [*] [(*function*)] ["*name*"] [–*count*]
{
[[*module.*] *line-no* | *variable* | *function*]
[IF *condition*]
|
IF *condition*
}
[{*commands* [; *commands* ...]}]

CALL *function* ([*arg* [, ...]])

CLEANUP [ALL]

CONTINUE [INTERRUPT | QUIT]

DATABASE *database-name*

DISABLE {*name* | *refno* | *function* | ALL}

DUMP [GLOBALS | ALL] [>>*filename*]

ENABLE {*name* | *refno* | *function* | ALL}

Escape: !*command*

EXIT

FUNCTIONS [*pattern*] [>>*filename*]

GROW [SOURCE | COMMAND] [-] *integer*

HELP [*command* | ALL]

Interrupt: {**CONTROL-C** | **DEL**}

LET *variable* = *expression*

LIST [BREAK] [TRACE] [DISPLAY]

NOBREAK {*name* | *refno* | *function* | ALL}

NOTRACE {*name* | *refno* | *function* | ALL}

SE PRINT *expression* [>>*filename*]

OL PRINT *expression*
[>>*filename* | PROGRAM = "*program*"]

READ *filename* [.4db]

Redraw: **CONTROL-R**

RUN[*arg* [*arg* ...]]

Screen: **CONTROL-P**

Search: {/ | ?}[*pattern*]

STEP [*n*] [INTO] [NOBREAK]

TIMEDELAY [SOURCE | COMMAND] *integer*

Toggle: **CONTROL-T**

TRACE [*] [(*function*)] ["*name*"]
{
[*module.*] *line-no*
|
variable
|
function
|
FUNCTIONS
}
[{*commands* [; *commands* ...]}] [>> *filename*]

TURN [ON | OFF]
{
AUTOTOGGLE
|
DISPLAYSTOPS
|
EXITSOURCE
|
PRINTDELAY
|
SOURCETRACE
} ...

USE [[=] *pathname* [, ...]]

VARIABLE [*variable* | GLOBALS | ALL]
[>> *filename*]

VIEW [*module* | *function*]

WHERE [>> *filename*]

WRITE [BREAK] [TRACE] [DISPLAY] [ALIASES] [>>]
[*filename*]

Default Function Key Aliases

Key	Debugger Command
F1	help
F2	step
F3	step into
F4	continue
F5	run
F6	list break trace
F7	list
F8	dump
F9	exit