OpenQM

2.12-4

Quick Reference Guide

Quick Reference Guide

Special thanks to:

Users of the OpenQM product who have contributed topics and suggestions for this manual.

Such information is always very much appreciated so please continue to send comments to support@openqm.com.

Table of Contents

1   VOC Record Types

D   Data definition
> F2 = Field number
> F3 = Conversion code
> F4 = Column heading
> F5 = Format code
> F6 = Single/multi-value flag
> F7 = Association name
> F8 = Available for user use. Not referenced by QM.

F   File reference
> F2 = Data pathname(s)
> F3 = Dictionary pathname
> F4 = Subfile names for a multifile
> F5 = File inclusion flags for ACCOUNT.SAVE and FILE.SAVE
> F6 = Special mode flags

K   Keyword
> F2 = Keyword number
> F3+ = Alternative expansion for use as a command

M   Menu
> F2 = Title
> F3 = Item text
> F4 = Action. End with semicolon for "Press return to continue" prompt
> F5 = Help text
> F6 = Access key
> F7 = Hide inaccessible entries?
> F8 = Access control subroutine name
> F9 = Prompt text
> F10 = Exit codes
> F11 = Stop codes

PA   Paragraph
> F2+ = Commands to execute

PH   Phrase
> F2 = Expansion

PQ   PROCs
> F2+ = Commands to execute

Q   Indirect file pointer
> F2 = Account name or path
> F3 = File name in account
> F4 = Server name (QMNet connections)

R   Remote command pointer
> F2 = File name
> F3 = Record name
> F4 = Security subroutine name (optional)

S   Sentence
> F2 = Command

V   Verb
> F2 = Type
> F3 = Item to execute
> F4 = Additional information
> F5 = Security subroutine name (optional)

X   Miscellaneous

User defined VOC record types can be parsed using the X-type $VOC.PARSER record:
> Field 2   Multivalued list of VOC record type codes
> Field 3   Corresponding multivalued list of catalogue names of handler programs.

The handler subroutine declaration is
```
SUBROUTINE handler.name(verb, voc.rec)
```

2   Dictionary Records

A   Pick style data definitions. A-types are provided for compatibility with other environments. D and I-type records should be used by preference. See the *QM Reference Manual* for details.

C   Calculated values. Similar in concept to I-types, these are programs that return a value via the @ANS variable. C-types are provided for compatibility with other environments and I-types should be used by preference. See the *QM Reference Manual* for details.

D   Direct data types. Describes data that is held in a field of the record.

I   Indirect data types. Describes data that can be evaluated from data in fields of the record by evaluating a QMBasic expression.

L   Links to other files. Used only in query processor commands.

PH   Phrases for use in query processor commands.

S   Pick style data definitions. S-types are provided for compatibility with other environments. D and I-type records should be used by preference. See the *QM Reference Manual* for details.

X   Other miscellaneous data.

D and I type records define data and share a common format:

ID	Name by which the field is to be known
Field 1	Type (D or I) plus optional description
Field 2	Field number (D-type) or expression (I-type)
Field 3	Conversion code for entry and display of this field
Field 4	Display name to be used by LIST or SORT when displaying this field
Field 5	Format in which data is to be displayed
Field 6	Single or multi-valued indicator (S or M)
Field 7	Association name
Field 8	Available for user use. Not referenced by QM.
Field 9-10	Reserved for internal use
Field 11	Available for user use. Not referenced by QM.
Field 12+	Reserved for internal use

Reserved Dictionary Records

@ID   A D-type record defining the record id.

@   A phrase record defining the default list of items to be displayed by **LIST** and **SORT** in the absence of any other field names.

@LPTR   A phrase record defining the default list of items to be displayed by **LIST** and **SORT** in the absence of any other field names when output is directed to a printer. If this record is not present, the query processor uses the @ record instead.

@MODIFY A phrase record defining the default list of items to be processed by the **MODIFY** command.

@SHOW A phrase record defining the default list of items to be displayed by **SHOW** in the absence of any other field names.

3   Command Editor

Dot Commands

Unless otherwise stated, *n* defaults to 1 if omitted.

.A*n text*	Append *text* to command stack entry *n*.
.C*n* /*old*/*new*/**G**	Change string *old* to *new* in stack entry *n*.
.D*n*	Delete stack entry *n*.
.D *name*	Delete VOC entry *name* if it is a sentence or paragraph record.
.I*n text*	Insert *text* as stack entry *n*.
.L*n*	List the most recent *n* commands. The value of *n* defaults to 20.
.L *name*	List VOC entry *name*.
.R*n*	Recall stack entry *n* to the top of the stack without deleting the original copy.
.R *name*	Read VOC entry *name* to the top of the stack if it is a sentence or paragraph.
.S *name n m*	Save stack lines *m* to *n* as VOC entry *name*.
.U*n*	Convert stack entry *n* to upper case.
.X*n*	Execute command *n*.
.X *file record*	Execute command stored in named file and record.
.?	Display help text.

Special keys

Ctrl-A or HOME	Move cursor to start of command.
Ctrl-B or Cursor Left	Move cursor left one place.
Ctrl-D or DELETE	Delete character under cursor.
Ctrl-E or END	Move cursor to end of command.
Ctrl-F or Cursor Right	Move cursor right one place.
Ctrl-G	Exit from the command stack and return to a clear command line.
Ctrl-K	Delete all to the right of the cursor.
Ctrl-N or Cursor Down	Display "next" command from command stack.
Ctrl-O or Insert	Toggle insert/overlay mode.
Ctrl-P or Ctrl-Z or Cursor Up	Display "previous" command from command stack.
Ctrl-R	Search back up the command stack for a given string.
Ctrl-T	Interchange characters before cursor.
Ctrl-U	Convert command to uppercase.
Backspace	Backspace one place.

4   Commands

* *text*
    Defines a comment in a paragraph.

$ECHO {**ON** | **OFF**}
    Controls paragraph tracing.

! *command*
    Executes a shell (operating system) command.

ABORT {*text*}
    Terminates all active processing and returns to the command prompt.

ACCOUNT.RESTORE {**BINARY**} {**DET.SUP**} {**DIRECTORY**} {**NO.CASE**}
    {**NO.INDEX**} {**NO.OBJECT**} {**POSITIONED**}
    Restores a Pick style ACCOUNT.SAVE tape.

ACCOUNT.SAVE {*account.name*} {**BINARY**} {**DET.SUP**}
    {**EXCLUDE.REMOTE** | **INCLUDE.REMOTE**}
    Creates a Pick style ACCOUNT.SAVE tape.

ADMIN
    System administration tools menu (QMSYS account only).

ADMIN.SERVER
    Manage security rules for QMNet server connections.

ADMIN.USER
    Performs management of the register of user names for network connections.

ALIAS
    Lists all defined command aliases.

ALIAS *command target*
    Sets command as an alias for *target*.

ALIAS *command*
    Removes alias for command.

ANALYSE.FILE {**DICT**} *file.name* {**STATISTICS**} {**NO.PAGE**} {**LPTR**}
ANALYZE.FILE {**DICT**} *file.name* {**STATISTICS**} {**NO.PAGE**} {**LPTR**}
    Reports information regarding the structure and efficiency of a file.

AUTHENTICATE *username password*
    Changes user authentication in a startup script.

AUTHKEY *password*
    Encrypts a password for use with **AUTHENTICATE**.

AUTOLOGOUT *period*
    Sets an inactivity period after which a process will automatically be logged out.

BASIC {*file.name*} {*record.name...* | *****} {*options*}
    Runs the QMBasic compiler. Options are:

CHANGED	Compile only if source has changed.
DEBUGGING	Enable debugging of the compiled program.
LISTING	Generate a compiler listing record with a .LIS suffix.
NO.PAGE	Suppress pagination
NO.QUERY	Suppress prompts (e.g. catalogued item replacement)
NOXREF	Omit cross reference tables from compiled program.
XREF	Generate a compiler listing record including a variable cross-reference table.

    An X-type record named $BASIC.OPTIONS in the named file or in the VOC can be used to apply compiler defaults. Fields 2 onwards contain any of the following options, one per field:

CATALOGUE {LOCAL	GLOBAL}	Catalogues program after compilation.
DEBUGGING	Compiles the program in debug mode.
DEFINE *name* {*value*}	Defines token name.
LISTING	Generates a listing record.
MODE *option.name*	Sets the given compilation mode.
NOXREF	Compiles the program with no cross reference tables.
XREF	Generate a compiler listing record including a variable cross-reference table.

BELL {**OFF** | **ON**}
    Disables or enables the audible alarm for many commands and programs.

BLOCK.PRINT *text*
Prints text on the default printer using large characters.

BLOCK.TERM *text*
    Displays text using large characters.

BREAK {**OFF** | **ON**}
    Disables or enables the break key.

BREAK CLEAR
    Clears deferred breaks.

BREAK COUNT
    Reports the number of active **BREAK OFF** commands.

BREAK ON USER *n*
    Enables the break key for the specified user.

BUILD.INDEX *filename* {*field(s)* | **ALL**} {**CONCURRENT**}
    Populates an alternate key index.

CATALOGUE {*file.name* {*catalogue.name*}} *record.name* {**LOCAL** | **GLOBAL**}
      {**NO.QUERY**} {**NOXREF**}
CATALOG {*file.name* {*catalogue.name*}} *record.name* {**LOCAL** | **GLOBAL**}
      {**NO.PAGE**} {**NO.QUERY**} {**NOXREF**} {**RELATIVE**}
    Adds a compiled QMBasic program to the catalogue.

LOCAL	Adds a VOC entry to point to the compiled program
GLOBAL	Adds to global catalogue for access from all accounts

Neither   Adds to private catalogue for access from current account only

CD *file.name* {*Itype.name*} ... {**NO.QUERY**}
Compiles I-type records in dictionaries.

CD ALL
Compiles all I-type records in all dictionaries referenced by F-type VOC entries in the account.

CD LOCAL
Compiles all I-type records in all dictionaries referenced by F-type VOC entries in the account that do not have a directory separator in the dictionary pathname.

CLEAN.ACCOUNT
Clears the $HOLD, $COMO and $SAVEDLISTS files.

CLEAR.ABORT
Clears the abort status so that an ON.ABORT paragraph can restart the application.

CLEAR.DATA
CLEARDATA
Clears the data queue created by the **DATA** command or the QMBasic **DATA** statement.

CLEAR.FILE {**DATA** | **DICT**} *file.name*
Deletes all records from a file.

CLEAR.INPUT
CLEARINPUT
Clears all unprocessed characters entered at the keyboard.

CLEAR.LOCKS {*lock.number*}
Releases task locks.

CLEARPROMPTS
Clears all stored inline prompts and responses.

CLEAR.SELECT {*list.number*}
CLEAR.SELECT ALL
CLEARSELECT {*list.number*}
CLEARSELECT ALL
Clears one or all active select lists.

CLEAR.STACK
Clears the command stack.

CLR
Clears the terminal screen.

CNAME *old.file.name***,** *new.file.name*
CNAME *old.file.name* **TO** *new.file.name*
Changes the name of a file.

CNAME {**DICT**} *file.name old.record.id***,** *new.record.id*
CNAME {**DICT**} *file.name old.record.id* **TO** *new.record.id*
Changes the name of record(s) within a file.

COMO ON *record.name*
  Commences recording of command output in named record of $COMO file.

COMO OFF
  Terminates recording of command output.

COMPILE.DICT *file.name* {*name*} ... {**NO.QUERY**} {**NO.PAGE**}
  Compiles A, C, I and S-type records in dictionaries.

COMPILE.DICT ALL {**NO.PAGE**}
  Compiles all A, C, I and S-type records in all dictionaries referenced by F-type VOC entries in the account.

COMPILE.DICT LOCAL {**NO.PAGE**}
  Compiles all A, C, I and S-type records in all dictionaries referenced by F-type VOC entries in the account that do not have a directory separator in the dictionary pathname.

CONFIG {**LPTR**}
  Reports licence details and configuration parameters.

CONFIG *param new.value*
  Modifies the value of a per-process configuration parameter.

CONFIGURE.FILE {**DICT**} *file.name parameters* {**REPORTING**}
  Changes the configuration of a file as defined by parameters:

DEFAULT	Resets all parameters to their default values.
DYNAMIC	Converts file to dynamic hashed type.
DIRECTORY	Converts file to directory type.
GROUP.SIZE *n*	Sets the group size in units of 1024 bytes (Range 1 to 8).
MINIMUM.MODULUS *n*	Sets the minimum modulus for the file.
LARGE.RECORD *bytes*	Sets the large record size in bytes.
SPLIT.LOAD *pct*	Sets the split load factor for the file.
MERGE.LOAD *pct*	Sets the merge load factor for the file.
NO.CASE	Converts to use case insensitive record ids.
NO.MAP	Suppress translation of reserved characters in record ids of directory files.
CASE	Converts to use case sensitive record ids.
NO.RESIZE	Disables dynamic file resizing.
RESIZE	Enables dynamic file resizing.
IMMEDIATE	Applies immediate file resizing.

  Parameters which are not specified retain their existing values.

COPY FROM {**DICT**} *src.file* {**TO** {**DICT**} *tgt.file*} {*src.rec*{**,***tgt.rec*}} {*options*}
  Copies selected records from one file to another, or within the same file. Options are:

ALL	Copy all records from *src.file* to *tgt.file*.
BINARY	Suppress mark translation when copying between hashed and directory file.
DELETING	Delete the record(s) from *src.file* after copying.
NO.QUERY	Suppresses confirmation prompt when using a select list.
OVERWRITING	Overwrites records that already exist in *tgt.file*.
REPORTING	Displays the id of each record as it is copied.
TEXT	Force text mode transfer when both files are directory files.
UPDATING	Only copies records if they already exist in the target file.

COPY.LIST *src.list* {**,** *tgt.list*} {**FROM** *src.file*} {**TO** *tgt.file*} {*options*}
    Copies a saved select list to another file or a different record in the same file. Options are:

CRT	Output the select list to the display. (Omit tgt.list and tgt.file).
DELETING	Delete *src.list* after copying.
LPTR {*n*}	Output the select list to logical print unit *n* (default 0).
NO.PAGE	With CRT, suppresses the pause between pages of output.
OVERWRITING	Allows overwriting of an existing list.

COPY.LISTP {{**DICT**} *src.file*} {*id.list* | ***} {*options*}
    Copies a saved select list to another file or a different record in the same file using Pick style syntax. Options are:

(B	Copy in binary mode between a hashed file and a directory file.
(D	Delete the record(s) from *src.file* after copying.
(I	Suppress display of record ids.
(N	Suppress pagination when displaying records on the terminal.
(O	Overwrite records that already exist in the target file.
(P	Send the record data to a printer.
(S	Suppress field numbers with (P or (T.
(T	Send the record data to the terminal.

COPYP {**DICT**} *src.file* {*id.list* | ***} {*options*}
    Copies selected records from one file to another, or within the same file using Pick style syntax. Options are:

(A	Force text mode transfer when both files are directory files.
(B	Copy in binary mode between a hashed file and a directory file.
(D	Delete the record(s) from *src.file* after copying.
(I	Suppress display of record ids.
(N	Suppress pagination when displaying records on the terminal.
(O	Overwrite records that already exist in the target file.
(P	Send the record data to a printer.
(S	Suppress field numbers with (P or (T.
(T	Send the record data to the terminal.

COUNT {**DICT**} *file.name*
      {**USING** {**DICT**} *file.name*}
      {*selection.clause*}
      {*record.id...*}
      {**FROM** *select.list.no*}
    Counts records meeting specified criteria.

CREATE.ACCOUNT *acc.name pathname* {**NO.QUERY**}
      {**MODE** *mmm*} {**USER** *uid*} {**GROUP** *gid*}
    Creates a new QM account.

CREATE.FILE {**DATA** | **DICT**} *file.name* {**DYNAMIC** | **DIRECTORY**} {*configuration*}
      {**USING DICT** *other.name*} {**ENCRYPT** *keyname*} {**NO.QUERY**}
      {**MODE** *ddd*{**:***sss*}} {**USER** *uid*} {**GROUP** *gid*}
    Creates a QM file. Configuration (dynamic files only) may be:

CASE	Creates a file using case sensitive record ids, overriding the CREATE.FILE.NO.CASE mode of the OPTION command.
PATHNAME path	Pathname of directory under which the file is to be created.
MINIMUM.MODULUS *n*	Sets the minimum modulus for the file.
GROUP.SIZE *size*	Sets the group size as a multiple of 1024 bytes (Range 1 - 8).

LARGE.RECORD *bytes*	Sets the large record size in bytes (default 80% of group size).
SPLIT.LOAD *pct*	Sets the split load factor for the file (default 80%).
MERGE.LOAD *pct*	Sets the merge load factor for the file (default 50%).
VERSION *vno*	Specifies internal file format for backward compatibility.
NO.CASE	Creates a file using case insensitive record ids.
NO.RESIZE	Creates the file with dynamic resizing disabled.

Other options are:

MODE *ddd*:*sss*	Sets access permissions (not Windows)
NO.MAP	Suppress translation of reserved characters in record ids of directory files.
USER *uid*	Sets the user owning the file (not Windows)
GROUP *gid*	Sets the group owning the file (not Windows)

CREATE.INDEX *filename field(s)* {**NO.NULLS**} {**PATHNAME** *index.path*}
>Creates an alternate key index

CREATE.KEY {*keyname* {*algorithm*} {*keystring*}}}
>Creates an encryption key.

CREATE.USER {*username* {*account*}}
>Creates a new user name in the register of users for network security checks.

CS
>Clears the terminal screen.

CT {**DICT**} *filename* {*record ...*} {**BINARY**} {**HEX**} {**LPTR** {*n*}} {**NO.QUERY**}
>Displays the content of record(s) from a file.

DATA {*text*}
>Used in paragraphs to supply data to a command or program instead of prompting the user.

DATE
>Displays the current day, date and time (e.g. "Wednesday, 18 February 2004 04:57pm")

DATE *date*
>Converts internal day number to date or vice versa.

DATE INTERNAL
>Displays the current date as an internal form day number.

DATE.FORMAT {**OFF** | **ON**}
>Turns on or off default European date format.

DATE.FORMAT DISPLAY
>Displays the default date format.

DATE.FORMAT INFORM
>Sets @SYSTEM.RETURN.CODE according to the default date format.

DATE.FORMAT *conv.code*
>Sets the default date conversion code.

DEBUG {*file.name*} *record.name* {**LPTR**} {**NO.PAGE**}
>Enters the QMBasic program debugger.

DELETE {**DICT**} *file.name* {*record.name ...* | **ALL**} {**NO.QUERY**}
Deletes specified records from a file.

DELETE.ACCOUNT *acc.name* {**FORCE**}
Deletes a QM account.

DELETE.CATALOGUE {*name*} {**GLOBAL** | **LOCAL**}
DELETE.CATALOG {*name*} {**GLOBAL** | **LOCAL**}
Removes an entry from the system catalogue.

DELETE.COMMON {*name* | **ALL**}
Deletes one or all named common blocks.

DELETE.DEMO
Deletes files that form the demonstration database.

DELETE.FILE {**DATA** | **DICT**} *file.name* {**FORCE**} {**NO.QUERY**}
Deletes one or both portions of a file.

DELETE.INDEX *file.name* {*field(s)* | **ALL**}
Deletes one or more alternate key indices.

DELETE.KEY {*keyname*}
Deletes an encryption key.

DELETE.LIST *list.name*
Deletes a previously saved select list.

DELETE.PRIVATE.SERVER *name*
Deletes a QMNet private server definition.

DELETE.SERVER *name*
Deletes a QMNet server definition.

DELETE.USER {*username*}
Deletes a user name from the register of users for network security checks.

DISABLE.INDEX *filename* {*field(s)* | **ALL**}
Disables update and use of one or more alternate key indices.

DISABLE.PULISHING
Disables publishing to all replication subscribers.

DISPLAY *text* {**:**}
DISPLAY @(*col*,*row*)*text* {**:**}
Displays text at the user's terminal.

DUMP {**DICT**} *filename* {*record ...*} {**LPTR** {*n*}} {**NO.QUERY**}
Displays the content of record(s) from a file in hexadecimal and character format.

ECHO {**OFF** | **ON**}
Suspends or enables echoing of keyboard input.

ED {**DICT**} *file.name* {*record.id...*} {**NO.QUERY**}
    Enters the QM line editor.

EDIT {**DICT**} *file.name* {*record.id...*} {**NO.QUERY**}
    Enters the QM line editor (synonym for **ED**).

EDIT.CONFIG
    Edits QM configuration parameters.

EDIT.LIST {*list.name*}
    Invokes the **ED** line editor to edit a saved select list in the $SAVEDLISTS file.

ENABLE.PULISHING
    Reenables publishing to all replication subscribers.

ENCRYPT.FILE *filename field,keyname ...*
    Applies field level encryption to the named field(s).

ENCRYPT.FILE *filename keyname*
    Applies record level encryption to the named file.

FILE.SAVE {*account.list*} {**BINARY**} {**DET.SUP**}
    {**EXCLUDE.REMOTE** | **INCLUDE.REMOTE**} {**NO.QUERY**}
    Creates a Pick style FILE.SAVE tape.

FILE.STAT {**DICT**} {*account.list* | **ALL**} {**LPTR**}
    Creates a summary report of the files in one or more accounts.

FILE.STAT {{**DICT**} {*file.list*}} {**LPTR**}
    Creates a summary report of the files in the current account.

FIND.ACCOUNT *account.name*
    Locates an account on a Pick style FILE.SAVE tape.

FIND.PROGRAM *name*
    Locates and reports details of a catalogued program.

FORMAT {*file.name*} {*record.name*} {**CASE**}
    Reformats QMBasic source programs to aid readability.

FORM.LIST {**DICT**} *file.name record.id* {**TO** *list.no*}
    Creates an active select list from a list of record keys in a file.

FSTAT {**ON** | **OFF**} *file.name...*
    Enables or disables statistics collection for one or more named files.

FSTAT *file.name...* {**LPTR**}
    Reports statistics for the named files.

FSTAT GLOBAL {**LPTR**}
    Reports global system statistics.

FSTAT RESET
    Clears global statistics counters.

FSTAT
    Periodic global statistics display.

GENERATE *file.name*
    Generates a QMBasic include record from a dictionary.

GET.LIST *list.name* {**TO** *list.no*} {**COUNT.SUP**}
    Restores a previously saved select list.

GET.STACK {*stack.na*me}
    Restores a previously saved command stack.

GO *label*{**:**}
    Used within paragraphs to jump to a labelled line.

GRANT.KEY *keyname* {**GROUP**} *name ...*
    Grants access to an encryption key.

HELP
    Provides help on a wide variety of topics.

HSM {**OFF** | **ON** | **DISPLAY**} {**USER** *n*}
    Controls the Hot Spot Monitor performance monitoring tool.

HUSH {**OFF** | **ON**}
    Suspends or enables output to the display.

IF *value.1 rel.op value.2* **THEN** *sentence*
IF EXISTS *filename id* **THEN** *sentence*
IF NOT.EXISTS *filename id* **THEN** *sentence*
    Allows conditional execution of sentences within paragraphs.

INHIBIT.LOGIN {**OFF** | **ON**}
    Enables or disables QM login.

LIST {**DICT**} *file.name*
      {**USING** {**DICT**} *file.name*}
      {*field.name* {*field.qualifier*} *...*}
      {*selection.clause*}
      {*sort.clause*}
      {*display.clause*}
      {*record.id...*}
      {**FROM** *select.list.no*}
    Displays or prints a report from a database file.

LIST.COMMON
    Lists named common blocks.

LIST.DIFF *list1* {*list2* {*tgt.list*}} {**COUNT.SUP**}
    Forms the difference of two saved select lists.

LIST.FILES {**BRIEF**}
>   Displays statistics regarding the number of open files.

LIST.FILES DETAIL {{**DICT**} *filename*}
>   Displays details of open files.

LIST.FILES USER {*n*}
>   Displays pathnames of open files for the specified user.

LIST.INDEX *filename field(s)* {**STATISTICS** | **DETAIL**} {**LPTR** {*n*}} {**NO.PAGE**}
>   Reports details of one or more alternate key indices in the named file.

LIST.INDEX ALL {**LOCAL**} {**STATISTICS** | **DETAIL**} {**LPTR** {*n*}} {**NO.PAGE**}
>   Reports details of alternate key indices in all files.

LIST.INTER *list1* {*list2* {*tgt.list*}} {**COUNT**.SUP}
>   Forms the intersection of two saved select lists.

LIST.ITEM {**DICT**} *file.name*
>   {**USING** {**DICT**} *file.name*}
>   {*selection.clause*}
>   {*sort.clause*}
>   {*record.id...*}
>   {**FROM** *select.list.no*}
>   Displays or prints a report from a database file in internal format.

LIST.KEYS {**LPTR** {*unit*}}
>   Reports all encryption keys defined in the key vault.

LIST.KEYS *filename* {**LPTR** {*unit*}}
>   Reports encryption keys used by the named file.

LIST.LABEL {**DICT**} *file.name*
>   {**USING** {**DICT**} *file.name*}
>   {*field.name* {*field.qualifier*} *...*}
>   {*selection.clause*}
>   {*sort.clause*}
>   {*display.clause*}
>   {*record.id...*}
>   {**FROM** *select.list.no*}
>   Displays or prints a report from a database file as a page of labels.

LIST.LOCKS
>   Reports the state of the 64 system wide task locks.

LIST.PHANTOMS
>   Shows all currently active phantom processes.

LIST.READU {*user*} {**DETAIL**} {**NO.PAGE**} {**LPTR** {*n*}} {**WAIT**}
>   Displays details of file and record locks.

LIST.REPLICATED
Displays a summary of files that use replication.

LIST.SERVERS {**ALL**} {**DETAIL**} {**LPTR** {*n*}}
Displays a list of all defined QMNet servers.

LIST.TRIGGERS
Displays a summary of trigger functions used in an account.

LIST.UNION *list1* {*list2* {*tgt.list*}} {**COUNT.SUP**}
Forms the union of two saved select lists.

LIST.USERS
Lists users from the register of users for network security checks.

LIST.VARS {**ALL**}
Displays user @-variables.

LISTDICT *filename* {**LPTR**}
Lists a dictionary in Pick style format.

LISTF {**LPTR**}
Lists all files defined in the VOC.

LISTFL {**LPTR**}
Lists all files defined in the VOC that are local to the account.

LISTFR {**LPTR**}
Lists all files defined in the VOC that are remote to the account.

LISTK {**LPTR**}
Lists all keywords defined in the VOC.

LISTM {**LPTR**}
Lists all menus defined in the VOC.

LISTPA {**LPTR**}
Lists all paragraphs defined in the VOC.

LISTPH {**LPTR**}
Lists all phrases defined in the VOC.

LISTPQ {**LPTR**}
Lists all PROCs defined in the VOC.

LISTQ {**LPTR**}
Lists all indirect file references defined in the VOC.

LISTR {**LPTR**}
Lists all remote items defined in the VOC.

LISTS {**LPTR**}
Lists all sentences defined in the VOC.

LISTU {**DETAIL**}
Lists the users currently in QM.

LISTV {**LPTR**}
Lists all verbs defined in the VOC.

LOAD.**LANGUAGE** *pathname*
Install non-English message texts (QMSYS account only).

LOCK *lock.number* {**NO.WAIT**}
Sets a task lock.

LOGIN.PORT *port* {*account*} {*params*}
Login a serial port from within another QM session.

LOGOUT {*user*}
Terminates a QM process.

LOGOUT ALL
Terminates all QM processes except the one issuing the command. Restricted to administrators.

LOGTO *name* {**RESET** {**ALL**}}
Moves to an alternative account directory without leaving QM.

LOOP
 sentence(s)
REPEAT
Defines the top and bottom of a group of sentences to be repeated within a paragraph.

MAKE.INDEX *filename* *field(s)* {**NO.NULLS**} {**PATHNAME** *index.path*} {
CONCURRENT}
Creates and builds an alternate key index

MAP {**ALL**} {**LPTR** {*n*}} {**FILE** {*file.name*}}
Produces a map of the system catalogue.

MED {*file.name* {*menu.name*}}
Creates or modifies a menu definition.

MERGE.LIST *list1* *rel.op* *list2* {**TO** *tgt.list*} {**COUNT.SUP**}
Creates a new active select list by merging two other lists. *Rel.op* may be:
    INTERSECTION   Only those record keys that appear in both *list1* and *list2*.
    UNION   All record keys from both *list1* and *list2*.
    DIFFERENCE   All record keys from *list1* except those that are also in *list2*.

MESSAGE *user* {**IMMEDIATE**} {*message.text*}
Sends a message to selected other users.

MESSAGE OFF
Disables reception of messages.

MESSAGE ON
    Enables reception of messages.

MODIFY {**DICT**} *file.name* {*field list*}
    Enters the QM record modification processor.

NLS {*key*}
    Report national language support settings.

NLS DEFAULT
    Sets default national language support settings.

NLS *key value*
    Sets value for a named national language support settings. *Key* is:

CURRENCY	Currency symbol (maximum 8 characters)
THOUSANDS	Thousands separator character
DECIMAL	Decimal separator character
IMPLICIT.DECIMAL	Decimal separator character for implicit conversions

NSELECT {**DICT**} *file* {**FROM** *from.list*} {**TO** *to.list*}
    Refines a select list by removing items that are in a named file.

OFF   (Synonym for QUIT)
    Terminates the current QM session.

OPTION {*option.name* {**ON** | **OFF** | **DISPLAY** | **LPTR** {*unit*} } }
    Sets, clears or displays configurable options.

AMPM.UPCASE	Output am/pm suffix of some time conversions in uppercase.
ASSOC.UNASSOC.MV	Treat all multi-valued fields with no association as associated.
BACKSLASH.NOT.QUOTE	Causes the command processor to treat backslashes as data characters instead of being recognised as string quotes.
CHAIN.KEEP.COMMON	Retains the unnamed common block on use of CHAIN.
CHAINED.SELECT	If set, a query processor select operation that selects at least one record will execute any command in the DATA queue on completion. If no records are selected, the data queue is cleared.
CORRELATIVE.NOCASE	Causes string operations in correlative expressions to be performed in a case insensitive manner.
CORRELATIVE.REUSE	Causes all operators in correlative expressions to behave as though the reuse (R) flag was present.
CRDB.UPCASE	Output cr/db suffix of some decimal conversions in uppercase.
CREATE.FILE.NO.CASE	Causes CREATE.FILE to create files with case insensitive record ids by default.
DEBUG.REBIND.KEYS	Causes the QMBasic debugger to rebind the function keys on entry, replacing any user defined bindings with those specified in the terminfo entry for the current terminal type.

DIR.DTM	If enabled, writes to a directory file or closing a sequential file that has been written will update the date/time modified of the directory. (Not Windows)
DIV.ZERO.WARNING	Treat divide by zero as a warning, using a zero result.
DUMP.ON.ERROR	Causes generation of a process dump file at a process abort such as a run time fatal error.
ED.NO.QUERY.FD	Suppresses the confirmation prompt in the ED editor when using the FD command or its synonym DELETE.
FORCE.RELOAD	Force reload of QMBasic object code at compilation or catalogue.
INHERIT	Phantom processes will inherit the option settings of the parent process. Use of an OPTION command in the MASTER.LOGIN or LOGIN paragraphs of the phantom process may modify these settings.
INHERIT.OWNERSHIP	Causes CREATE.FILE to inherit ownership (user id and group id) from the account unless overridden by command options. This option has no effect on Windows systems.
KEEP.FILENAME.CASE	Causes CREATE.FILE to preserve the casing of the QM name of the file when creating the operating system directories that represent this file.
KEEP.OLD.OBJECT	Causes the QMBasic compiler to retain any previous version of the compiled object code if a compilation fails.
LOCK.BEEP	Emits a beep at the terminal once per second while waiting for a record or file lock.
NO.DATE.WRAPPING	Do not roll dates with overlarge day numbers into the next month on input conversion.
NO.SEL.LIST.QUERY	Suppresses display of the confirmation prompt in commands that take an optional select list of records to process. This is equivalent to use of the NO.QUERY option to those commands.
NO.USER.ABORTS	Suppress all options that allow a user to generate an abort event.
NON.NUMERIC.WARNING	Treat use of a non-numeric value as a number as a warning.
PAGINATE.ON.HEADING	When used with the RUN.NO.PAGE option, setting a page heading or page footing will cause output pagination to be enabled. The NO.PAGE option to the RUN command can be used to override this.
PICK.BREAKPOINT	Recognise Pick style BREAK.ON and BREAK.SUP clauses.
PICK.BREAKPOINT.U	Handle the U breakpoint option similarly to Pick.
PICK.EXPLODE	When using BY.EXP, if an associated field has only one value, do not explode this field.
PICK.GRAND.TOTAL	Show GRAND.TOTAL text on same line as values.
PICK.IMPLIED.EQ	Treats a field name immediately followed by a literal string enclosed in double quotes in a selection clause as though there was an EQ operator.
PICK.ML.CONV.MASK	Allows parenthesised format masks in the ML and MR conversion codes, disabling recognition of a left

	parenthesis as requesting negative values to be output in round brackets.
PICK.NULL	ML and MR conversions and numeric formats return null for null data rather than zero.
PICK.PROC	D3 compatibility for PQ style Procs. Use of this option implies the PROC.A option. Note that the features covered by this option may change in future releases.
PICK.WILDCARD	Recognise Pick style wildcards in equality tests.
PROC.A	Causes the Proc A(n,m) command not to terminate copying data at the end of the field.
QUALIFIED.DISPLAY	Recognise Pick style qualified display clauses.
QUERY.MERGE.PRINT	Causes a query report directed to a printer to merge the output as continuation of the active print job if the printer is already active, leaving it active on completion of the report. Without this option, the printer is closed at the end of the report.
QUERY.NO.CASE	Causes the query processor to handle selection operations and sorts in a case insensitive manner.
QUERY.PRIORITY.AND	Causes the AND operator to take priority over the OR operator in query processor commands.
QUERY.STRING.COMP	Forces all EQ and NE operator comparisons in query processor selection clauses to be performed as string comparisons regardless of the data type.
RUN.NO.PAGE	Causes the RUN and DEBUG commands to start the program with screen pagination disabled. This option also affects user catalogued programs. If the PAGINATE.ON.HEADING option is set, pagination will be enabled if the application sets a page heading or page footing unless the NO.PAGE option has been used when starting the program.
SELECT.KEEP.CASE	Causes QM to preserve the case of record ids when building a select list from a directory file on an operating system that uses case insensitive file names. This currently only affects Windows systems.
SHOW.STACK.ON.ERROR	Displays the call stack at a fatal program error, showing the program name, line number (where available), and object code address.
SPACE.MCT	Modifies the behaviour of the MCT conversion code such that only the first character and letters immediately after a space are converted to uppercase.
SPOOL.COMMAND	Displays the operating system command used to perform printing (Diagnostic aid, not Windows).
STACKED.ACCOUNT	Causes a QMBasic EXECUTE statement that switches accounts to revert to the previous account on return to the calling program.
SUPPRESS.ABORT.MSG	Suppresses display of program location diagnostic information when a QMBasic ABORT.QMB.ABORT statement is executed.
UNASS.WARNING	Treat unassigned variables in QMBasic programs as a

warning.

WITH.IMPLIES.OR   Imply an OR rather than AND between multiple WITH
                               clauses.

The option command also supports short forms that set multiple options as described in the *QM Reference Manual*.

PASSWORD {*username*}
Changes the password for a QM user on Windows 95, 98 and ME.

PAUSE
Displays a "Press return to continue" prompt.

PDEBUG {*command*}
Runs the phantom debugger.

PHANTOM {**DATA**} {**NO.LOG**}{**USER** *n*} *command*
Starts execution of a verb, sentence or paragraph as a background process.

PRINTER *print.unit* **AT** *printer.name*
Directs printed output to the named printer.

PRINTER *print.unit* **BOTTOM.MARGIN** *n*
Sets the bottom margin size.

PRINTER *print.unit* **CLOSE**
Closes a print unit, overriding any previous use of the KEEP.OPEN option (see SETPTR).

PRINTER *print.unit* **FILE** *filename recordname*
Selects the destination for printed output.

PRINTER *print.unit* **KEEP.OPEN**
Allows merging of successive printer output into a single print job.

PRINTER *print.unit* **LEFT.MARGIN** *n*
Sets the left margin size.

PRINTER *print.unit* **LINES** *n*
Sets the number of lines per page.

PRINTER *print.unit* **QUERY**
Reports the current settings of the width, lines per page, top margin, bottom margin and left margin

PRINTER *print.unit* **RESET**
Resets to the default values for width, lines per page, top margin, bottom margin and left margin.

PRINTER *print.unit* **TOP.MARGIN** *n*
Sets the top margin size.

PRINTER *print.unit* **WIDTH** *n*
Sets the number of characters per line.

PSTAT {{**USER**} *userno*} {**LEVEL** *level*}
 The PSTAT command displays the status of one or all QM processes. *Level* is formed by adding the following components:
    1   Report each program and subroutine in the call stack. If not included, only the currently active program is reported.
    2   Report internal subroutine calls within each reported program and subroutine. If not included, only external subroutine calls are reported.

PTERM BINARY {**ON** | **OFF**}
 Enables or disables binary transmission mode for terminal data.

PTERM BREAK {**ON** | **OFF**}
 Determines whether the break key is treated as special or as a normal data character.

PTERM BREAK { *n* | **^c** }
 Specifies the character to be recognised as the break key (not supported in all terminal emulators).

PTERM CASE {**INVERT** | **NOINVERT**}
 Controls case inversion of alphabetic characters received from the keyboard.

PTERM DISPLAY
 Displays terminal settings.

PTERM LPTR
 Reports terminal settings on the default printer.

PTERM MARK {**ON** | **OFF**}
 Determines whether characters 28-30 are translated to field, value and subvalue marks on entry from the keyboard.

PTERM NEWLINE {**CR** | **LF** | **CRLF** }
 Determines the newline sequence sent in terminal output.

PTERM PROMPT "*string*" {"*string*" }
 Changes the command prompt.

PTERM RESET *string*
 Sets the terminal reset string sent on return to the command prompt.

PTERM RETURN {**CR** | **LF** }
 Determines the value returned by KEYIN() and related functions when the return key is pressed.

PUBLISH {**DICT**} *src.file* {**DICT**} *server:account:tgt.file* ...
 Set up a file for replication.

PUBLISH {**DICT**} {*src.file* | **ALL**} QUERY
 Query replication status.

PUBLISH.ACCOUNT {**DICT**} *server:account*
 Select files for publication from displayed list

PUBLISH.ACCOUNT CANCEL *server:account*
>   Select files to cancel for publication from displayed list

QSELECT {**DICT**} *filename* {*id* | ***** | **FROM** *list*} {**TO** *list*} {**SAVING** *field*}
>   Constructs a select list from the content of selected records.

QUIT
>   Terminates the current QM session.

REBUILD.ALL.INDICES {**CONCURRENT**}
>   Rebuilds all alternate key indices in an account.

REDO {*interval* {*count*}} *command*
>   Repeat a command at specified intervals.

REFORMAT {**DICT**} *file.name*
>   {**USING** {**DICT**} *file.name*}
>   {*field.name* {*field.qualifier*} *...*}
>   {*selection.clause*}
>   {*sort.clause*}
>   {*display.clause*}
>   {*record.id...*}
>   {**FROM** *select.list.no*}
>   {**TO** *new.file.name*}
>   Constructs a new file from data in the source file.

RELEASE *filename id...*
>   Release specified record locks.

RELEASE FILELOCK *filename*
>   Release file lock.

RENAME *old.file.name***,** *new.file.name*   (Synonym for CNAME)
RENAME *old.file.name* **TO** *new.file.name*
>   Changes the name of a file.

RENAME {**DICT**} *file.name old.record.id***,** *new.record.id*
RENAME {**DICT**} *file.name old.record.id* **TO** *new.record.id*
>   Changes the name of record(s) within a file.

REPORT.SRC {**OFF** | **ON**}
>   Controls display of the @SYSTEM.RETURN.CODE variable on return to the command
>   prompt.

REPORT.STYLE
>   Displays the default query processor report style.

REPORT.STYLE *name*
>   Sets the default query processor report style.

REPORT.STYLE OFF
>   Disables the default query processor report style.

RESET.MASER.KEY
Resets the master encryption key after relicensing or moving an encrypted system.

RESTORE.ACCOUNTS {*target*} {**BINARY**} {**DET.SUP**} {**NO.CASE**}
{**NO.INDEX**} {**NO.OBJECT**} {**POSITIONED**}
Restores all accounts from a Pick style FILE.SAVE tape.

REVOKE.KEY *keyname* {**GROUP**} *name ...*
Removes access to an encryption key.

RUN {*file.name*} *record.name* {**LPTR**} {**NO.PAGE**}
Starts execution of a compiled QMBasic program or a VOC style record stored in an alternative
file.

SAVE.LIST *list.name* {**FROM** *list.no*} {**COUNT.SUP**}
Saves an active select list for future use.

SAVE.STACK {*stack.name*}
Saves the current command stack.

SCRB {*screen.file*} {*screen.name*}
Runs the screen builder to create or modify a screen definition for use by the QMBasic
!SCREEN() subroutine.

SEARCH {**DICT**} *file.name*
{**USING** {**DICT**} *file.name*}
{*selection.clause*}
{*sort.clause*}
{*record.id...*}
{**FOR** *string1 string2...*}
{**FROM** *select.list.no*}
{**SAVING** {**UNIQUE**} *field.name* {**NO.NULLS**}}
{**STRINGS** *file.name record.id*}
{**TO** *select.list.no*}
Constructs a list of records based on text matching.

SECURITY
Reports the current security setting.

SECURITY {**OFF** | **ON**}
Disables or enables QM's internal security system.

SED {**DICT**} *file.name* {*record.id*}
Enters the QM full screen editor.

SEL.RESTORE {**DICT**} *target.file.name* {*item.li*st} {**BINARY**} {**DET.SUP**}
{**NO.CASE**} {**NO.INDEX**} {**NO.OBJECT**} {**POSITIONED**}
Restores a single file from a Pick style ACCOUNT.SAVE or FILE.SAVE tape.

SELECT {**DICT**} *file.name*
　　　{**USING** {**DICT**} *file.name*}
　　　{*selection.clause*}
　　　{*sort.clause*}
　　　{*record.id...*}
　　　{**FROM** *select.list.no*}
　　　{**SAVING** {**UNIQUE**} {**MULTI.VALUE**} *field.name* {**NO.NULLS**}}
　　　{**TO** *select.list.no*}
　Constructs a select list.

SELECTINDEX {**DICT**} *filename indexname* {**TO** *list*} {**COUNT.SUP**}
　Constructs a select list of all indexed values from an alternate key index.

SET *variable value*
SET *variable* **EVAL** *expression*
　Sets a value into an @-variable.

SET.DEVICE *device.name* {*format*} {**NO.QUERY**}
　Opens a tape or pseudo-tape for processing by QM.

SET.ENCRYPTION.KEY.NAME *filename fieldname,keyname ...*
　Updates the encryption key name associated with one or more fields.

SET.ENCRYPTION.KEY.NAME *filename keyname*
　Updates the encryption key name used for record level encryption.

SET.EXIT.STATUS *value*
　Sets the final exit status returned by QM to the operating system.

SET.FILE {*account* { *filename* { *pointer*} } }
SET.FILE {*server:account* { *filename* { *pointer*} } }
　Adds a Q-pointer to the VOC to reference a remote file.

SET.LANGUAGE *language.c*ode
　Select message text language.

SET.PRIVATE.SERVER *name ip.address*{*:port*} *username password*
　Defines a QMNet private server.

SET.QUEUE *queue* {**,** *width***,** *depth***,** *top.margin***,** *bottom.margin***,** *mode* {**,** *options* }}
　Sets the characteristics of a form queue. The options are as for SETPTR.

SET.SERVER *name ip.address*{*:port*} *username password* {**NO.QUERY**}
　Defines a QMNet server.

SET.TRIGGER *file.name*
　Displays the trigger function associated with a dynamic file.

SET.TRIGGER *file.name function.name* {*modes*}
　Sets the trigger function associated with a QM data file.

SET.TRIGGER *file.name* **""**
　Removes the trigger function associated with a QM data file.

SETPORT *port* {**BAUD** *rate*} {**BITS** *bits.per.byte*} {**PARITY** *parity*} {**STOP.BITS** *stop*}
{**BRIEF**}
Sets communications parameters for a serial port.

SETPTR *unit* {**,** *width***,** *depth***,** *top.margin***,** *bottom.margin***,** *mode* {**,** *options* }}
Sets print unit characteristics. The mode values are:

1	Direct output to the underlying operating system print manager.
3	Save output in the $HOLD file.
4	Direct output to the stderr (standard error) file unit.
5	Direct output to the terminal auxiliary printer port.
6	Directs output to a file and prints it.

The options available are:

AS { NEXT } { *id* }	Specifies the hold file record name in modes 3 and 6.
AS PATHNAME *path*	Specifies a destination pathname for output in modes 3 and 6.
AT *printer.name*	Specifies the printer name in modes 1 and 6.
BANNER *text*	Sets the text to appear on a banner page.
BRIEF	Suppresses the normal confirmation prompt.
COPIES *n*	Specifies the number of copies to be printed.
EJECT	Appends a form feed to the end of the print data.
FORM.FEED *mode*	Determines the form feed sequence used by the QMBasic PRINT statement. The *mode* may be CRFF (default) or FF.
GDI	Specifies that the Windows GDI mode API calls are to be used.
INFORM	Displays file details when opening a print file in mode 3.
KEEP.OPEN	Keeps the printer open to merge successive printer output.
LANDSCAPE	Specifies that the output is to be printed in landscape format.
LEFT.MARGIN *n*	Inserts a margin of *n* spaces to the left of the printed data.
NEWLINE *mode*	Determines the newline sequence used by the QMBasic PRINT statement. The *mode* may be CR, LF (default) or CRLF.
NFMT	Specifies that no page formatting is to be applied to the output data.
NODEFAULT	Leaves omitted options at their current value.
NOEJECT	Suppresses the normal page throw at the end of a print job (Windows only).
OPTIONS *xxx*	Passes the given option(s) to the underlying operating system print spooler (e.g. OPTIONS "landscape" on a Linux system).
OVERLAY *subr*	Identifies a catalogued subroutine for use in generating a graphical overlay for each page of output.
PCL	Specifies that this printer supports PCL.
PORTRAIT	Specifies that the output is to be printed in portrait format.
PREFIX *path*	Prefixes output with the contents of the named file.
RAW	Specifies that the Windows non-GDI mode API calls are to be used.
SPOOLER *name*	Selects a non-default spooler on Linux and FreeBSD.
STYLE *name*	Sets the default query processor report style for reports directed to this print unit.

Additional options available with PCL are shown below. In many cases, the list of acceptable parameter values can be extended by modifying the SYSCOM $PCLDATA record.

CPI *n*	Specifies the number of characters per inch when used with PCL. The value may be non-integer.
DUPLEX	Selects duplex (double sided) printing, binding on the long edge. This option is currently only supported in conjunction

	with the PCL option.
DUPLEX SHORT	Selects duplex (double sided) printing, binding on the short edge. This option is currently only supported in conjunction with the PCL option.
LPI *n*	Specifies the number of lines per inch when used with PCL. The value must be 1, 2, 3, 4, 6, 8, 12, 16, 24 or 48.
PAPER.SIZE *xx*	Specifies the paper size when used with PCL. Valid size names are A4, LETTER, LEGAL, LEDGER, A3, MONARCH, COM_10, DL, C5, B5.
SYMBOL.SET *xx*	Specifies the character set when used with PCL. Valid values of *xx* are ROMAN8 (the default), LATIN1, ASCII, PC8.
WEIGHT *xx*	Specifies the font weight when used with PCL. Valid values of *xx* are ULTRA-THIN, EXTRA-THIN, THIN, EXTRA-LIGHT, LIGHT, DEMI-LIGHT, SEMI-LIGHT, MEDIUM, SEMI-BOLD, DEMI-BOLD, BOLD, EXTRA-BOLD, BLACK, EXTRA-BLACK, ULTRA-BLACK though specific printers might not support all values.

SETPTR DISPLAY {**LPTR** {*printer*}}
> Reports settings of all print units.

SETPTR *unit*, **DISPLAY**
> Reports settings of the specified print unit in a form that can be captured by a program and used later to restore the printer settings.

SETUP.DEMO {**UPDATING**}
> Creates, resets or updates files that form the demonstration database.

SH
> Executes an interactive shell. (Not currently available on Windows)

SH *command*
> Executes a shell (operating system) command.

SHOW {**DICT**} *file.name*
>   {**USING** {**DICT**} *file.name*}
>   {*field.name* {*field.qualifier*} ...}
>   {*selection.clause*}
>   {*sort.clause*}
>   {*display.clause*}
>   {*record.id...*}
>   {**FROM** *select.list.no*}
>   {**TO** *select.list.no*}
>
> Provides an interactive means of building select lists

SLEEP *time*
> Suspends execution of further commands until a given number of seconds have elapsed or until a specified time of day.

SORT {**DICT**} *file.name*
      {**USING** {**DICT**} *file.name*}
      {*field.name* {*field.qualifier*} *...*}
      {*selection.clause*}
      {*sort.clause*}
      {*display.clause*}
      {*record.id...*}
      {**FROM** *select.list.no*}
  Displays or prints a report from a database file, sorted by record id.

SORT.ITEM {**DICT**} *file.name*
      {**USING** {**DICT**} *file.name*}
      {*selection.clause*}
      {*sort.clause*}
      {*record.id...*}
      {**FROM** *select.list.no*}
  Displays or prints a report from a database file, sorted by record id in internal format.

SORT.LABEL {**DICT**} *file.name*
      {**USING** {**DICT**} *file.name*}
      {*field.name* {*field.qualifier*} *...*}
      {*selection.clause*}
      {*sort.clause*}
      {*display.clause*}
      {*record.id...*}
      {**FROM** *select.list.no*}
  Displays or prints a sorted report from a database file as a page of labels.

SORT.LIST *src.list* {**TO** *tgt.list*} {*sort.rule*}
  Sorts a saved select list.

SP.ASSIGN *options*
  Sets output to a Pick style form queue. Options are:
      n   number of copies to print
      F*qno*   specifies the form queue number
      H   directs output to the hold file
      O   keeps the print unit open until PRINTER CLOSE is used
      Q*qno*   same as **F***qno*
      R*unit*   uses the specified print unit, default 0
      S   suppresses printing

SP.CLOSE
  Resets the "keep open" state of the default printer and closes the active print job.

SP.OPEN
  Sets the "keep open" state of the default printer, merging successive print requests into a single job.

SP.VIEW {*file*} {*record*} {**LPTR** *n*}
  View and, optionally, print records from $HOLD or other files.

SPOOL *file record(s)* {**LINES** *m n*} {**LNUM**} {**LPTR** *n*}
  Sends specified records to the printer.

SREFORMAT {**DICT**} *file.name*
>   {**USING** {**DICT**} *file.name*}
>   {*field.name* {*field.qualifier*} *...*}
>   {*selection.clause*}
>   {*sort.clause*}
>   {*display.clause*}
>   {*record.id...*}
>   {**FROM** *select.list.no*}
>   {**TO** *new.file.name*}

Constructs a new file from data in the source file, in record id order.

SSELECT {**DICT**} *file.name*
>   {**USING** {**DICT**} *file.name*}
>   {*selection.clause*}
>   {*sort.clause*}
>   {*record.id...*}
>   {**FROM** *select.list.no*}
>   {**SAVING** {**UNIQUE**} {**MULTI.VALUE**} *field.name* {**NO.NULLS**}}
>   {**TO** *select.list.no*}

Constructs a sorted select list.

STATUS { **ALL** }

Displays a list of active phantom processes.

STOP

Terminates the currently active paragraph.

SUBSCRIBE *server*{**:***port*} *username password* {*options*}

Start replication subscriber phantom.

SUM {**DICT**} *file.name*
>   {**USING** {**DICT**} *file.name*}
>   *field.name* {*field.qualifier*} *...*
>   {*selection.clause*}
>   {*record.id...*}
>   {**FROM** *select.list.no*}

Displays a summary report showing only the total values of specified fields.

T.ATT *device.name* {*format*} {**NO.QUERY**}

Opens a tape or pseudo-tape for processing by QM.

T.DET

Detach a previously assigned tape device.

T.DUMP {**DICT**} *filename* {*id...*} {**BINARY**} {**COUNT.SUP**} {**DET.SUP**} {**FROM** *listno*}

Saves data to a Pick style T-DUMP tape.

T.EOD

Position a tape device to the end of the recorded data.

T.FWD

Move a tape device forward by one file.

T.LOAD {**DICT**} *filename* {*item.list*} {**BINARY**} {**COUNT.SUP**} {**DET.SUP**} {
OVERWRITING}
   Restores data from a Pick style T-DUMP tape.

T.RDLBL
   Read the label from a tape device.

T.READ
   Read data from a tape device.

T.REW
   Rewind a tape device.

T.STAT
   Report the status of a tape device.

T.WEOF
   Write end of file marker to a tape device.

TERM
   Displays the terminal screen layout dimensions.

TERM { *columns* } { **,** *lines* } { *term.type*}
   Sets the terminal screen layout dimensions.

TERM COLOUR { *bgc*} { **,** *fgc*}
   Sets the background and foreground colours.

TERM DEFAULT
   Sets the terminal screen layout dimensions to their default values.

TERM DISPLAY
   Displays input key codes and output control sequences for the currently selected terminal.

TIME
   Displays the current date and time (e.g. 16:57:00 18 Feb 2004).

TIME *time*
   Converts and external format time to its internal value or vice versa.

TIME INTERNAL
   Displays the current time in internal form (seconds since midnight).

UMASK {*rights*}
   Set access rights to be applied on file creation.

UNLOCK {**USER** *user.no*} {**FILE** *file.no*} {**ALL** | *record.ids...*}
   Releases specified record locks set by any process.

UNLOCK {**USER** *user.no*} {**FILE** *file.no*} **FILELOCK**
   Releases specified file locks set by any process.

UNLOCK TASKLOCK *lock.no...*
> Releases specified task locks set by any process.

UNLOCK.KEY.VAULT
> Enables user access to encryption keys when using key vault security.

UPDATE.ACCOUNT
> Copies all system VOC entries from NEWVOC, setting the correct locations for system files.

UPDATE.LICENCE
> Applies new licence details.

UPDATE.RECORD {**DICT**} *file* {**USING** {**DICT**} *dict*}
>   {**FROM** *listno* | **ALL** | *id...* | **INQUIRING** {*prompt*}}
>   *field***,***value* {**CONV "***spec***"**} { *field***,***value ...*}
>   **DELETING** *field*
>   {**COUNT.SUP**}
>   {**VERIFY.SUP**}
>   {**EXCLUSIVE**}
>   {**WAIT** | **NO.WAIT**}
>   {**CREATING**}
>   {**OVERWRITING**}
>   {**REPORTING**}
>   {**LPTR** {*n*}}
>   {**NO.PAGE**}
> Performs batch mode updates to specified records.

UPDATE.RECORD {**DICT**} *file* {**USING** {**DICT**} *dict*}
>   {**FROM** *listno* | **ALL** | *id...* | **INQUIRING** {*prompt*}}
>   {**ID.SUP**}
>   {**COL.SUP**}
> Performs interactive updates to specified records.

WHO
> Displays the current user number and account name.

WHERE
> Displays the pathname of the current account.

5   Inline Prompts

<<{control,} text {, check}>>

control   determines the way in which the prompt is displayed and how it is actioned on subsequent execution of the same statement or another with the same text.

@(*col***,** *row***)**	Specifies the display position for the prompt text.
@(BELL)	Sounds the audible warning unless suppressed by BELL OFF.
@(CLR)	Clears the display.
@(TOF)	Positions to the top left of the display.
A	Always prompt.
C*n*	Returns the *n*'th token of the current sentence.
C*n***:***default*	Returns the *n*'th token of the current sentence of default if none.
C*m***-***n*	Returns tokens *m* to *n* of the current sentence.
C*n***+**	Returns tokens *n* onwards of the current sentence.
C#	Returns the number of tokens in the current sentence.
CHANGE(*str***,***old***,***new***)**	Change all occurrences of *old* to *new* in *str*.
F(*file***,** *key* **{,***field* **{,** *value* **{,** *subvalue*}}})**	Fetch data from a database record.
ICONV(*str***,***code***)**	Apply input conversion *code* to *str*.
I*n*	Returns the *n*'th word of the current sentence, prompting if null.
L*n*	Returns the next item from select list *n*.
OCONV(*str***,***code***)**	Apply input conversion *code* to *str*.
R	Prompts repeatedly, separating items by a space.
R(*string***)**	Prompts repeatedly, separating items by string.
S*n*	Returns the *n*'th word of the command entered at the command prompt.
SUBR(*name***)**	Executes catalogued QMBasic subroutine *name*, returning the result.
SUBR(*name***,***arg***)**	Executes catalogued QMBasic subroutine *name*, passing in the given argument(s) and returning the result.
SYSTEM(*n***)**	Returns the value of the QMBasic **SYSTEM(***n***)** function.
U	Converts data entered in response to the prompt to uppercase.
@*var*	Retrieves the value of the given variable.
@*var***:***default*	Retrieves the value of the given variable, returning *default* if none.
$*var*	Retrieves the value of an operating system environment variable.
$*var***:***default*	Retrieves the value of an operating system environment variable, returning *default* if none.

text   is the prompt text to be displayed. An equals sign is automatically added to the end of

the prompt text.

check is used to check whether the response to the prompt is valid. This may be a either a pattern match template or an input conversion code. Conversion codes must be enclosed in round brackets.

6   Query Processor Keywords

General form of a display clause element:

Prefix	Data Item	Suffix
AVG	D-type item	CONV "*code*"
PCT {*n*}	I-type item	FMT "*spec*"
TOTAL	EVAL "*expr*" {AS *xx*}	COL.HDG "*text*"
MAX		ASSOC "*name*"
MIN		ASSOC.WITH *field*
BREAK.ON {"*text*"}		DISPLAY.LIKE *field*
BREAK.SUP {"*text*"}		SINGLE.VALUE
ENUM		MULTI.VALUE
CALC		NO.NULLS

Relational and Logical Operators

=	EQ	EQUAL		
#	NE	<>	><	NOT
>	GT	GREATER	AFTER	
<	LT	LESS	BEFORE	
>=	GE	=<		
<=	LE	=>		
BETWEEN				
AND	OR			
OR	!			
LIKE	MATCHES	MATCHING		
UNLIKE	NOT.MATCHING			
SAID	SPOKEN	~		
IN				
NOT.IN				

The **NO.CASE** qualifier can be used after a relational operator to make the comparison case insensitive.

Pattern Matching Elements

...	Zero or more characters of any type
0X	Zero or more characters of any type
*n*X	Exactly *n* characters of any type
*n-m*X	Between *n* and *m* characters of any type
0A	Zero or more alphabetic characters
*n*A	Exactly *n* alphabetic characters
*n-m*A	Between *n* and *m* alphabetic characters
0N	Zero or more numeric characters
*n*N	Exactly *n* numeric characters
*n-m*N	Between *n* and *m* numeric characters
"*string*"	A literal string which must match exactly.

ABSENT.IGNORE
Ignore absent records, suppressing the display of such records on completion of the command.

ABSENT.NULL
Treats an absent record as a null item rather than an error.

ALL.MATCH
Used in **SEARCH** to specify that the records must contain all of the given search strings.

field.name {*field.qualifier*} **AS** *synonym*
Defines a synonym for a field and any qualifying information.

field.name **ASSOC "***name***"**
Specifies that the field is to be treated as part of a named association.

field.name **ASSOC.WITH** *name*
Specifies that the field is to be associated with some other named field.

AVERAGE *field* {*field.qualifiers*} {**NO.NULLS**}
AVG *field* {*field.qualifiers*} {**NO.NULLS**}
Shows the average value of field at the end of the report and at breakpoints

BOXED
Generates a boxed report on a PCL printer.

BREAK.ON { **"***options***"** } *field*
Generates a breakpoint whenever the field value changes. Control codes in options are:

B{*n*}	Start a new page, retaining the field value for inclusion in the page heading/footing.
D	Omit the subtotal line if there is only one line of detail for this breakpoint.
L	Emit a blank line in place of the breakpoint.
N	Resets page number to one at each breakpoint. This implies the P option if not used with B or P.
O	Only show the field value on the first detail line within the breakpoint.
P	Start a new page.
U	Toggles production of a line of hyphens above any subtotals, etc.
V	Print the breakpoint field value.

BREAK.SUP { **"***options***"** } *field*
>   Generates a breakpoint whenever the field value changes but does not show the field as a column in the report. Control codes in options are as for **BREAK.ON**.

BY {**NO.CASE**} *field*
>   Sorts records into ascending order of field prior to display or when building a select list.

BY.DSND {**NO.CASE**} *field*
BY-DSND {**NO.CASE**} *field*
>   Sorts records into descending order of field prior to display or when building a select list.

BY.EXP {**NO.CASE**} *field* {*explosion limiter*}
BY-EXP {**NO.CASE**} *field* {*explosion limiter*}
>   Sorts records into ascending order of field, exploding multivalued items.

BY.EXP.DSND {**NO.CASE**} *field* {*explosion limiter*}
BY-EXP-DSND {**NO.CASE**} *field* {*explosion limiter*}
>   Sorts records into descending order of field, exploding multivalued items.

CALC *field*
>   Prefixes an I-type field name or an evaluated expression, causing the calculation to be performed on the total lines using accumulated values from the detail lines.

CAPTION "*text*"
>   Specifies text to appear at the left edge of the grand total line with **AVERAGE**, **ENUM**, **MAX**, **MIN**, **PERCENT** or **TOTAL**.

field **COL.HDG** *text*
>   Defines an alternative column heading for reported data.

COL.HDG.ID
>   Uses the displayed field names as the default column headings in a report.

COL.HDR.SUPP
COL-HDR-SUPP
COL.HDR.SUP
COL-HDR-SUP
>   Suppresses page and column headings.

COL.SPACES *n*
COL.SPCS *n*
>   Determines the number of spaces inserted between columns of a tabular report.

COL.SUP
COL-SUPP
>   Suppresses column headings. With **LIST.ITEM** and **SORT.ITEM**, this suppresses line numbering.

field **CONV** *conv.spec*
>   Defines an alternative conversion for reported data.

COUNT.SUP
>   Suppresses display of the number of records listed or selected at the end of the command.

CSV {*mode*} {**"***delimiter***"**} {**TO** *pathname* | **AS** *file id* {**NO.QUERY** | **APPENDING**}}
    Specifies a comma separated report.

CUMULATIVE *field* {*field.qualifiers*}
    Shows a cumulative value of the named field.

DBL.SPC
DBL-SPC
    Inserts a blank line between records in a tabular report.

DEFAULT
    Parses the query using default option settings.

DELIMITER "*string***"** {**TO** *pathname* | **AS** *file id* {**NO.QUERY** | **APPENDING**}}
    Specifies the separating character(s) to be used in a delimited report.

DET.SUP
    Suppresses reporting of detail lines, leaving only page and column headers, totals, footers and
    the final record count.

field.name **DISPLAY.LIKE** *other.field*
    Displays the field using the attributes of another field defined in the dictionary.

field **DISPLAY.NAME** *text*
    Defines an alternative column heading for reported data.

ENUM *field* {*field.qualifiers*} {**NO.NULLS**}
ENUMERATE *field* {*field.qualifiers*} {**NO.NULLS**}
    Shows a count of values of field at the end of the report and at breakpoints

EVAL *expr*
EVALUATE *expr*
    Evaluates an expression as though it were an I-type defined in the dictionary.

FIRST {*n*}
    Selects of displays only the first n records meeting any supplied selection criteria.

field **FMT** *fmt.spec*
    Defines an alternative format for reported data.

FORCE
    Forces display of headings in empty report.

FOOTER "*text***"**
FOOTING "*text***"**
    Defines a page footing for the report. The following control codes can be included in text:
        B{*n*}   Inserts data from the corresponding B control code in a **BREAK.ON** or
                BREAK.SUP.
        C   Centres the current line of the footing text. When used with G, the C option centres the
                current element of the text.
        D   Inserts the date.
        F{*n*}   Inserts the file name in a field of *n* spaces. If *n* is omitted, a variable width is used.

G   Inserts spaces to expand the text to the width of the output device.
H*n*   Sets horizontal position (column), numbered from 1.
I{*n*}   Inserts the record id in a field of *n* spaces. If *n* is omitted, a variable width is used.
L   Inserts a new line at this point in the text.
N   Suppresses pagination of the output to the display.
O   Reverses the elements separated by G tokens in the current line on even numbered pages. This is of use when printing double sided reports.
P{*n*}   Inserts the page number, right justified in *n* spaces. If omitted, *n* defaults to 4.
R{*n*}   Same as I{*n*}.
S{*n*}   Inserts the page number, left justified in *n* spaces. If omitted, *n* defaults to 1.
T   Inserts the time and date.

FROM *list.no*
Specifies the select list to be used as a source of record ids for processing by the query.

FOR *string1 string2 ...*
Specifies strings for the **SEARCH** command.

GRAND.TOTAL "*text***"**
GRAND-TOTAL "*text***"**
Specifies text to appear at the left edge of the grand total line with **AVERAGE**, **ENUM**, **MAX**, **MIN**, **PERCENT** or **TOTAL**.

HDR.SUP
HDR-SUPP
Suppresses the default page heading in a query.

HEADER "*text***"**
HEADING "*text***"**
Defines a page footing for the report. The following control codes can be included in text:
B{*n*}   Inserts data from the corresponding B control code in a **BREAK.ON** or **BREAK.SUP**.
C   Centres the current line of the footing text. When used with G, the C option centres the current element of the text.
D   Inserts the date.
F{*n*}   Inserts the file name in a field of *n* spaces. If *n* is omitted, a variable width is used.
G   Inserts spaces to expand the text to the width of the output device.
H*n*   Sets horizontal position (column), numbered from 1.
I{*n*}   Inserts the record id in a field of *n* spaces. If *n* is omitted, a variable width is used.
L   Inserts a new line at this point in the text.
N   Suppresses pagination of the output to the display.
O   Reverses the elements separated by G tokens in the current line on even numbered pages. This is of use when printing double sided reports.
P{*n*}   Inserts the page number, right justified in *n* spaces. If omitted, *n* defaults to 4.
R{*n*}   Same as I{*n*}.
S{*n*}   Inserts the page number, left justified in *n* spaces. If omitted, *n* defaults to 1.
T   Inserts the time and date.

ID.ONLY
Causes the query processor to ignore the default listing phrase and show only record ids.

ID.SUP
ID-SUPP
Omits the default display of @ID from the report.

field **IN** {**NO.CASE**} *listname*
Includes only records where the content of *field* is in the saved select list identified by *listname*.

field **IN** {**NO.CASE**} "*filename id*"
Includes only records where the content of *field* is in record *id* of *filename*.

LABEL *template.name*
LABEL NO.DEFAULT
Specifies the label template record name for **LIST.LABEL** and **SORT.LABEL**.

field **LIKE** {**NO.CASE**} *template*
Compares field against template, selecting items matching the template.

LOCKING
Takes a file lock on the file being processed, preventing updates during the report.

LPTR { *unit* }
Directs the output of the query to a printer.

MARGIN *width*
Specifies the width of a blank left margin to appear in the report output.

field **MATCHES** *template*
field **MATCHING** *template*
Compares field against template, selecting items matching the template.

MAX *field* {*field.qualifiers*}
Shows the maximum value of field at the end of the report and at breakpoints

MIN *field* {*field.qualifiers*} {**NO.NULLS**}
Shows the minimum value of field at the end of the report and at breakpoints

field **MULTI.VALUE**
field **MULTIVALUED**
Forces the field to be processed as a multi-valued item.

NEW.PAGE
Causes each record in a report to start on a new page.

NO.CASE
Used in **SEARCH** to specify that case insensitive string comparison is to be used.

NO.GRAND.TOTAL
Suppresses the grand total line.

NO.INDEX
Causes the query processor to ignore any alternate key index.

NO.MATCH
Used in **SEARCH** to specify that the records must contain none of the given search strings.

NO.NULLS
Suppresses null items. Used with the **AVERAGE**, **ENUMERATE**, **MIN** or **SAVING**.

NO.PAGE
NOPAGE
Suppresses the normal page end prompt.

NO.SPLIT
Causes the query processor to avoid splitting records across pages where possible.

field **NOT.IN** {**NO.CASE**} *listname*
Includes only records where the content of *field* is not in the saved select list identified by *listname*.

field **NOT.IN** {**NO.CASE**} "*filename id*"
Includes only records where the content of *field* is not in record *id* of *filename*.

field **NOT.MATCHING** *template*
Compares field against template, selecting items not matching the template.

ONLY
Causes the query processor to ignore the default listing phrase and show only record ids.

OVERLAY *subr.name*
Sets a graphical page overlay.

PAGESEQ *filename id*
Specifies a control record to ensure contiguous page numbering of separate reports.

PAN
Used in reports directed to the display, permits the total width of the report to exceed that of the display.

PERCENTAGE {*dp*} *field* {*field.qualifiers*}
PERCENT {*dp*} *field* {*field.qualifiers*}
PCT {*dp*} *field* {*field.qualifiers*}
% {*dp*} *field* {*field.qualifiers*}
Reports field as a percentage of the total of the value of the field in all selected records.

REPEATING
Causes single valued data to be repeated against each value in other fields.

REQUIRE.INDEX
Terminates the query unless it can make use of an alternate key index.

REQUIRE.SELECT
Terminates the query if there is no active select list.

field **SAID** *value*
field **SPOKEN** *value*
field **~** *value*
> Compares field against value using Soundex phonetic matching.

SAMPLE {*n*}
> Selects or displays only the first *n* records meeting any supplied selection criteria.

SAMPLED {*n*}
> Processes every *n*'th record.

SAVING {**UNIQUE**} {**MULTI.VALUE**} *field.name* {**NO.NULLS**}
> Used in **SELECT** or **SSELECT** to save the content of a field instead of the record id.
>
> | | |
> |---|---|
> | UNIQUE | Omits duplicate in the saved list. |
> | MULTI.VALUE | Expands values and subvalues as separate list entries. |
> | NO.NULLS | Omits null items from the saved list. |

SCROLL
> Used in a report directed to the display, enables scrolling back through report pages.

field **SINGLE.VALUE**
field **SINGLEVALUED**
> Forces the field to be processed as a single-valued item.

STRINGS *filename id*
> Specifies a control record containing the search strings for **SEARCH**.

STYLE *name*
> Sets the query processor report style.

SUPP
> Suppresses page headings.

TO *list.no*
> Used in **SELECT**, **SSELECT** or **SEARCH**, specifies the select list to be created.

TO *new.file.name*
> Used in **REFORMAT**, specifies the name of the output file.

TOTAL *field* {*field.qualifiers*}
> Shows the total value of field at the end of the report and at breakpoints

field **UNLIKE** *template*
> Compares field against template, selecting items not matching the template.

USING {**DICT**} *file.name*
> Uses the specified item in place of the dictionary of the file being processed.

VERT {**FMT "***format***"**}
VERTICALLY {**FMT "***format***"**}
> Produces a vertical format report.

WHEN *condition*
  Introduces a selection clause for a multi-valued field.

WITH {**EVERY**} *condition* {*rel.op* {**EVERY**} *condition...*}
  Introduces a selection clause.

WITH NO *field*
  Tests whether *field* is null.

WITHOUT *field*
  Tests whether *field* is null.

7   @-Variables and Constants

@AM	Attribute mark (synonym for @FM)
@FM	Field mark
@IM	Item mark
@SM	Subvalue mark
@SVM	Subvalue mark (synonym for @SM)
@TM	Text mark
@VM	Value mark
@FALSE	0
@TRUE	1

Variables

Except where indicated, these items are read-only

@ABORT.CODE	The cause of execution of the last abort. Values are: 0 No abort has occurred 1 A QMBasic **ABORT** statement or the **ABORT** command has been used. 2 The Abort option has been selected after the break key was pressed. 3 An internal error has occurred.
@ABORT.MESSAGE	The text message associated with the most recent abort event.
@ACCOUNT	Synonym for @WHO.
@ANS	Contains the result of the last virtual attribute expression evaluated. This variable can be updated, usually only in C-type dictionary items.
@COMMAND	The last command entered at the command prompt or initiated using the QMBasic **EXECUTE** statement.
@COMMAND.STACK	The history of commands executed at the command prompt. The most recent command is field 1.
@CONV	Extended information for user defined conversion codes.
@CRTHIGH	The number of lines per page of the display.
@CRTWIDE	The width of the display.
@DATA	Data from **REFORMAT** command.
@DATA.PENDING	The data on the **DATA** queue, if any.
@DATE	The internal format date value at which the last command started execution.
@DAY	The day of the month at which the last command started execution as a two digit value.

@DICTRECS	The query processor sets this variable to an item mark delimited copy of the dictionary records that were used to construct the display clause elements of the query (including any item with the BREAK.SUP prefix). For dictionary records that contain object code (C/I types and A/S types with correlatives), the object code is omitted.
@DS	The operating system specific directory delimiter character, / on Linux or FreeBSD, \ on Windows.
@FILE.NAME	The name of the file referenced in the most recent query processor command.
@FILENAME	Synonym for @FILE.NAME
@FMT	The query processor sets this variable to a field mark delimited list of the width and justification codes for each item in the display clause of the query (including any item with the BREAK.SUP prefix).
@GID	User's group id number for all platforms except Windows. Same as **SYSTEM(29)**.
@HOSTNAME	The name of the server computer system. Same as **SYSTEM(1015)**.
@ID	The record id of the record being processed by a query processor command or an I-type function.
@IP.ADDR	The IP address associated with a network user. Same as **SYSTEM(42)**.
@ITYPE.MODE	The mode of execution of an I-type. It has three possible values: 0 Normal 1 Evaluation of the old index value when updating or deleting a record from a file with an alternate key index. 2 Evaluation of the new index value when updating or adding a record to a file with an alternate key index.
@LEVEL	The current command processor depth (**EXECUTE** level). The initial command processor is level one, each **EXECUTE** level increments this by one.
@LOGNAME	User's login name. On Windows, this is converted to uppercase.
@LPTRHIGH	The number of lines per page of print unit zero. Depending on the current setting of the **PRINTER** flag, this may refer to the display or to the printer.
@LPTRWIDE	The width of print unit zero. Depending on the current setting of the **PRINTER** flag, this may refer to the display or to the printer.
@MONTH	The month in which the last command started execution as a two digit value.
@NB	Break number level. Set to zero on detail lines and one upwards on break lines. A value of 255 represents the grand total line.
@NI	Item counter. Used in I-types, this holds the number of records retrieved by the query processor command.
@OPTION	Contains a copy of field 4 of the V-type VOC entry when a verb

	starts execution. Use of this variable enables related commands to be handled by a single program.
@PARASENTENCE	The sentence that invoked the most recent paragraph or sentence.
@PATH	The pathname of the current account.
@PIB	The PROC primary input buffer.
@POB	The PROC primary output buffer.
@QM.GROUP	The QM user group to which the user running this session belongs, a null string if none.
@QMSYS	The pathname of the system account.
@RECORD	The data of the record being processed by an I-type function.
@SELECTED	The total record count for the most recent **SELECT** or **SSELECT** operation. A QMBasic **SELECT** operation against a dynamic file processes the file one group at a time and this variable will show the record count for the group being processed.
@SENTENCE	The currently active sentence. This is different from @COMMAND if the command runs a paragraph, sentence or menu.
@SEQNO	Hold file sequence number for most recent print job using uniquely sequenced file numbers.
@SIB	The PROC secondary input buffer.
@SOB	The PROC secondary output buffer.
@SYSTEM.RETURN.CODE	A status value returned from most commands.
@SYS.BELL	Initially contains the ASCII BEL character (character 7). The **BELL OFF** command changes @SYS.BELL to a null string and BELL ON reverts to the default character.
@TERM.TYPE	Terminal type.
@TIME	The internal format time value (seconds since midnight) at which the last command started execution.
@TRANSACTION.ID	The unique id number for the currently active transaction. Zero if no transaction is active. Same as **SYSTEM(1007)**.
@TRANSACTION.LEVEL	The transaction depth. Zero when no transaction is active, incremented for each active transaction, decremented when a transaction terminates. Same as **SYSTEM(1008)**.
@TRIGGER.RETURN.CODE	A status value returned set by trigger functions that return a **STATUS()** value of ER$TRIGGER.
@TTY	Terminal device name. This variable is provided for compatibility with other systems. It contains one of the following values: console QMConsole interactive session on Windows /dev/... QMConsole interactive session on Linux or FreeBSD telnet Telnet session phantom Phantom process port Serial port connection vbsrvr QMClient process

	Other process types may be added in future.
@UID	User's user id number for all platforms except Windows. Same as **SYSTEM(27)**.
@USER	Synonym for @LOGNAME.
@USER0 to @USER4	These variables are initially set to zero and may be updated by QMBasic programs to provide status information, etc. QM places no rules on the use of these variables and does not update them at any time..
@USERNO	User number.
@USER.NO	Synonym for @USERNO.
@USER.RETURN.CODE	This variable is initially set to zero and may be updated by QMBasic programs to provide status information, etc. QM places no rules on the use of this variable and does not update it at any time.
@VOC	A file variable for the VOC which can be used in place of opening it explicitly within user written application code.
@WHO	User's account name.
@YEAR	The last two digits of the year in which the last command started execution.
@YEAR4	The four digit year number in which the last command started execution.

8 QMBasic Compiler Directives

Directives may be written using a # character in place of the leading $ character.

$CATALOGUE {*name*} {**GLOBAL** | **LOCAL**} {**NO.QUERY**}
Add program to the system catalogue after successful compilation.

$DEBUG
Compile the program in debug mode.

$DEFINE *name value*
Associate a value with a symbolic name at compile time.

$ELSE
Used with $IFDEF or $IFNDEF to select alternative source statements.

$ENDIF
Terminates a conditional compilation element started with $IFDEF or $IFNDEF.

$EXECUTE "command**"**
Executes a QM command during program compilation.

$IFDEF *name*
Compile the following statements only if name is defined.

$IFNDEF *name*
Compile the following statements only if name is not defined.

$INCLUDE {*filename* } *record.id*
Include text from another record. Include records may be in either directory or dynamic files.

$INSERT {*filename* } *record.id*
Synonym for $INCLUDE.

$LIST {**ON** | **OFF**}
Start, suspend and resume generation of a listing of the program and any associated error messages.

$MODE *option* {**,** *option...*}
Sets or resets language options for improved compatibility with other multi-value databases. Options are:

DEFAULT	Turn off all options.
CASE.SENSITIVE	Enables case sensitivity for names of labels, variables and user defined functions.
COMPATIBLE.APPEND	Modifies the behaviour of the append modes of the S<f,v,sv> assignment operator, the **INS** statement and the **INSERT()** and **REPLACE()** functions to match that of other multivalue products.
COMPOSITE.READNEXT	Modifies how the **READNEXT** statement handles exploded select lists.

CONDITIONAL.STATEMENTS	Allow most statements that have a **THEN/ELSE** clause to be used as conditional elements in **WHILE** or **UNTIL**.
DEFAULT.UNASS.ARGS	Enables substitution of a default value for unassigned arguments in a **SUBROUTINE** or **FUNCTION** statement.
FOR.STORE.BEFORE.TEST	Stores the new value of the control variable in a **FOR**/**NEXT** construct before testing the end condition.
HEADING.NO.EJECT	Makes the **NO.EJECT** mode of the QMBasic **HEADING** statement the default, suppressing the automatic page throw on setting a new heading.
IMPLIED.STOP	Inserts an implied **STOP** instead of a **RETURN** at the end of a program, subroutine or function.
NO.ECHO.DATA	Suppresses echo of data for **INPUT** if it is taken from the **DATA** queue.
PICK.ENTER	Pick style processing of **ENTER**.
PICK.ERRMSG	Pick style syntax for **STOP** and **ABORT**.
PICK.JUMP.RANGE	Causes the **ON GOSUB** and **ON GOTO** statements to continue at the next statement if the index value is out of range.
PICK.MATRIX	Selects Pick style matrices which do not have a zero element and cannot be resized.
PICK.READ	Causes **READ**, **READL**, **READU**, **READV**, **READVU** and **READVL** statements to take on the Pick style behaviour in which the target variable is left unchanged if the record is not found.
PICK.SELECT	Changes the behaviour **of SELECTV** (and **SELECT** when **$MODE SELECTV** is in effect) when such that, if the default select list is active, that list is transferred into the target variable instead of building a new list. variable instead of building a new list.
PICK.SUBSTR	Causes substring assignment operations to take on the Pick style behaviour in which the variable is extended if the region to be overwritten is beyond the end of the current string value.
PRCLOSE.DEFAULT.0	**PRINTER CLOSE** defaults to printer 0 rather than all closing printers.
SELECTV	Changes the action of **SELECT** to be as for **SELECTV**.
STDFIL	Enables use of the default file variable.
STDFIL.SHARED	Enables use of the shared default file variable.
STRING.LOCATE	Numeric data in right aligned **LOCATE** is not treated as a special case.
TRAP.UNUSED	Displays a warning about variables that are assigned a value but never used.
UNASSIGNED.COMMON	Variables in common blocks are created unassigned instead of being initialised to zero.
UV.LOCATE	UniVerse Ideal / Reality flavour style **LOCATE**.

Prefixing a mode name (other than DEFAULT) with a minus sign turns off the named option. Default modes can be set using the $BASIC.OPTIONS record.

$NO.CATALOGUE

Indicates that the program should not be catalogued where the $BASIC.OPTIONS record includes the CATALOGUE option.

$NO.XREF

Indicates that the program should compiled without the cross-reference tables used for extended error diagnostics.

$NOCASE.STRINGS

Compiles the program with case insensitive string handling.

$PAGE

Inserts a page break in the compiler listing record.

$QMCALL

Makes the program available for calling via the QMClient QMCall function in systems running with the QMCLIENT configuration parameter set to 2.

9 QMBasic Statements and Functions

@(*col* {, *line*})
   Returns cursor control code to move to given column and line. Top left = 0,0.

@(*mode* {, *arg*})
   Sets specified terminal mode. Modes and their symbolic names are:

Mode	Token	Function	Argument
-1	IT$CS	Clear screen	
-2	IT$CAH	Cursor home	
-3	IT$CLEOS	Clear to end of screen	
-4	IT$CLEOL	Clear to end of line	
-5	IT$SBLINK	Start flashing text	
-6	IT$EBLINK	End flashing text	
-7	IT$SPA	Start protected area	
-8	IT$EPA	End protected area	
-9	IT$CUB	Backspace	No of characters (default 1)
-10	IT$CUU	Cursor up	No of lines (default 1)
-11	IT$SHALF	Start half brightness	
-12	IT$EHALF	End half brightness	
-13	IT$SREV	Start reverse video	
-14	IT$EREV	End reverse video	
-15	IT$SUL	Start underline	
-16	IT$EUL	End underline	
-17	IT$IL	Insert line	No of lines (default 1)
-18	IT$DL	Delete line	No of lines (default 1)
-19	IT$ICH	Insert character	No of characters (default 1)
-22	IT$DCH	Delete character	No of characters (default 1)
-23	IT$AUXON	Turn on printer	
-24	IT$AUXOFF	Turn off printer	
-29	IT$E80	Set 80 column mode	
-30	IT$E132	Set 132 column mode	
-31	IT$RIC	Reset inhibit cursor	
-32	IT$SIC	Inhibit cursor	
-33	IT$CUD	Cursor down	No of lines (default 1)
-34	IT$CUF	Cursor forward	No of characters (default 1)
-37	IT$FGC	Set foreground colour	Colour
-38	IT$BGC	Set background colour	Colour
-54	IT$SLT	Set line truncation	
-55	IT$RLT	Reset line truncation	
-58	IT$SBOLD	Set bold mode	
-59	IT$EBOLD	Reset bold mode	
-100 to -107		User definable via the u0 to u7 terminfo keys	
-108	IT$ACMD	Asynchronous command	Command to execute
-109	IT$SCMD	Synchronous command	Command to execute
-250	IT$STYLUS	PDA stylus control	Non-zero = enable
-251	IT$KEYS	PDA keyboard display	Non-zero = enable

   Colour values are
      0   IT$BLACK
      1   IT$BLUE
      2   IT$GREEN

3	IT$CYAN
4	IT$RED
5	IT$MAGENTA
6	IT$BROWN
7	IT$WHITE
8	IT$GREY
9	IT$BRIGHT.BLUE
10	IT$BRIGHT.GREEN
11	IT$BRIGHT.CYAN
12	IT$BRIGHT.RED
13	IT$BRIGHT.MAGENTA
14	IT$YELLOW
15	IT$BRIGHT.WHITE

ABORT {*msg*}
ABORTE {*msg.key*}
ABORTM {*msg.text*}
   Terminates the current program, returning to the command prompt. **ABORTE** interprets
   msg.key as a key to a message in the ERRMSG file. **ABORTM** treats *msg.text* as the actual
   message. **ABORT** normally behaves as **ABORTM** but can be switched to behave as
   ABORTE using the **$MODE** directive.

ABS(*expr***)**
   Returns absolute (positive) value of *expr*.

ABSS(*expr***)**
   Returns absolute (positive) value of each element of dynamic array *expr*.

ACCEPT.SOCKET.CONNECTION(*srvr.skt***,** *timeout***)**
   Opens a data socket on a server to handle an incoming connection.

ACOS(*expr***)**
   Returns the arc-cosine of *expr*. Angles are measured in degrees. If *expr* is a dynamic array, the
   function returns a similar dynamic array of results.

ALPHA(*string***)**
   Returns true (1) if *string* contains only alphabetic characters (A to Z, a to z). Returns false (0)
   for a null string or a string that contains non-alphabetic characters.

ANDS(*expr1***,** *expr2***)**
   Returns a dynamic array of results of logical AND operations between corresponding elements
   of dynamic arrays *expr1* and *expr2*.

ARG(*n***)**
   Returns argument *n* from the current subroutine. This is intended for use with subroutine
   declared with the VAR.ARGS option.

ARG.COUNT()
   Returns the number of arguments passed into a subroutine. This is intended for use with
   subroutine declared with the VAR.ARGS option.

ARG.PRESENT()
   Tests for presence of an argument variable in a subroutine or function declared with the

VAR.ARGS option.

ASCII(*expr***)**

Returns the ASCII equivalent of the supplied EBCDIC string. Characters that have no ASCII equivalent are returned as question marks.

ASIN(*expr***)**

Returns the arc-sine of *expr*. Angles are measured in degrees. If *expr* is a dynamic array, the function returns a similar dynamic array of results.

ASSIGNED(*var***)**

Tests whether a variable is assigned.

ATAN(*expr***)**

Returns the arc-tangent of *expr*. Angles are measured in degrees. If *expr* is a dynamic array, the function returns a similar dynamic array of results.

BEGIN TRANSACTION
 COMMIT / ROLLBACK
END TRANSACTION

Defines a transaction in which either all updates must be completed or discarded as a group.

BINDKEY(*key.string***,** *action***)**

Set, remove, query, save or restore key bindings.

BITAND(*expr1***,** *expr2***)**

Returns the bitwise AND of two integer values.

BITNOT(*expr***)**

Returns the bitwise inverse of an integer values.

BITOR(*expr1***,** *expr2***)**

Returns the bitwise OR of two integer values.

BITRESET(*expr***,** *bit***)**

Returns the value of *expr* with the specified bit set to zero. Bits are numbered from 0 to 31 from the least significant end of the value.

BITSET(*expr***,** *bit***)**

Returns the value of *expr* with the specified bit set to one. Bits are numbered from 0 to 31 from the least significant end of the value.

BITTEST(*expr***,** *bit***)**

Returns the value of the specified bit of *expr*. Bits are numbered from 0 to 31 from the least significant end of the value.

BITXOR(*expr1***,** *expr2***)**

Returns the bitwise exclusive or of two integer values.

BREAK {**KEY**} {**OFF** | **ON**}

Inhibits or enables break key handling. **BREAK OFF** decrements the inhibit counter. **BREAK ON** increments the inhibit counter. Breaks are inhibited if the counter is non-zero and will be handled when the counter is next set to zero.

BREAK {**KEY**} **CLEAR**
>   Clears any pending break event.

BREAK {**KEY**} *expr*
>   Equivalent to **BREAK OFF** if the value of *expr* is zero, **BREAK ON** if *expr* is positive and **BREAK CLEAR** if *expr* is negative.

CALL *name* {(*arg.list*)}
CALL @*var* {(*arg.list*)}
>   Calls the named subroutine. In the second form, variable *var* contains the name of the subroutine.

BEGIN CASE
>   **CASE** *expr*
>   >   *statement(s)*
>   **CASE** *expr*
>   >   *statement(s)*

END CASE
>   Executes the first set of conditioned statements for which *expr* is true. Use the pseudo conditional element **CASE 1** as the final test to include statements to be executed if none of the preceding conditions were true.

CATALOGUED(*name***)**
>   Test if name is in the system catalogue. Returns
>   >   0   the subroutine is not catalogued
>   >   1   the subroutine is catalogued locally as a V type VOC entry
>   >   2   the subroutine is catalogued privately
>   >   3   the subroutine is catalogued globally

CATS(*string1***,** *string2***)**
>   Returns the result of concatenating corresponding dynamic array components (fields, values and subvalues) from the supplied strings.

CHAIN *expr*
>   The current program terminates immediately, discarding local variables but retaining common variables. The command defined by *expr* is executed as though it replaced the sentence which invoked the program in which the **CHAIN** statement occurs. If this sentence is in a paragraph, the remainder of the paragraph will be executed when the **CHAIN**'ed program terminates.

CHANGE(*string***,** *old***,** *new*{**,** *occurrence*{**,** *start*}}**)**
>   Replaces the specified occurrences of *old* within string by *new*. *Occurrence* evaluates to the number of occurrences of *old* to be replaced. If omitted or specified as a value of less than one, all occurrences are replaced. *Start* specifies the first occurrence to be replaced. If omitted or specified as a value of less than one it defaults to one.

CHAR(*seq***)**
>   Returns a single character string containing the ASCII character with value *seq*. It is the inverse of the **SEQ()** function.

CHECKSUM(*data***)**
>   Returns a checksum value for the supplied data.

CHILD(*userno*)
Tests whether a phantom started by the current process is still running.

CLASS *name* {**MAX.ARGS** *limit*} {**INHERITS** *class.list*}
Defines a QMBasic class module for object oriented programming.

CLEAR
All local variables, including all elements of matrices, are set to zero. Files associated with local file variables will be closed. The value of variables in common areas are not affected.

CLEARCOMMON {*name*}
CLEAR COMMON {*name*}
All variables in the common block identified by *name* are set to zero, the unnamed common block if *name* is omitted. Other variables are not affected.

CLEARDATA
The data queue is cleared. Any keyboard type-ahead is not affected by this statement.

CLEARFILE *file.var* {**ON ERROR** *statement(s)*}
The file associated with the file variable is cleared, deleting all records from the file, contracting the file to its minimum modulus size and releasing disk space.

CLEARINPUT
CLEAR INPUT
Clears any type-ahead data. Data stored by the **DATA** statement is not affected.

CLEARSELECT {*list.no*}
CLEARSELECT ALL
CLEARSELECT *var*
Clears the specified (or all) select list.

CLOSE *file.var* {**ON ERROR** *statement(s)*}
Closes the file opened to *file.var*.

CLOSESEQ *file.var* {**ON ERROR** *statement(s)*}
Closes the sequential file opened to *file.var*.

CLOSE.SOCKET *skt*
Closes a socket.

COL1()
Used after a **FIELD()** function, returns the character position of the character immediately preceding the extracted substring.

COL2()
Used after a **FIELD()** function, returns the character position of the character immediately following the extracted substring.

COMMIT
Terminates a transaction, applying all cached updates to the database.

COMMON {*/name/*} *var1* {*,var2...*}
Defines variables as being in common blocks, memory areas that may be used to pass data

between different programs and subroutines.

COMPARE(*string1*, *string2* {, *justification*})

Compares two data items as character strings. Justification is "L" for left justified comparison or "R" for right justified comparison. If omitted or invalid, left justification is used. The function returns

  1   *string1* is greater than *string2*
  0   *string1* is equal to *string2*
  -1 *string1* is less than *string2*

COMPARES(*string1*, *string2* {, *justification*})

Compares each element of dynamic array *string1* with *string2*. Justification is "L" for left justified comparison or "R" for right justified comparison. If omitted or invalid, left justification is used. The function returns a dynamic array with the same structure as *string1* where each element is

  1   element of *string1* is greater than *string2*
  0   element of *string1* is equal to *string2*
  -1 element of *string1* is less than *string2*

CONFIG(*param*)

Retrieves the value of configuration parameters defined in the QM configuration file. If *param* is not recognised as a value configuration parameter name, the function returns zero.

CONNECT.PORT(*port*, *baud*, *parity*, *bits*, *stop*)

Converts a phantom process into an interactive session using the specified port as its terminal device.

CONTINUE

Equivalent to a jump to the **REPEAT** or **NEXT** statement of the innermost loop structure.

CONVERT *from.string* **TO** *to.string* **IN** *source.string*
CONVERT(*from.string*, *to.string*, *source.string*)

Replaces all occurrences of characters in *from.string* with their equivalent characters from *to.string* in *source.string*. If *to.string* is shorter than *from.string*, characters that have no substitutes are removed from *source.string*.

The first form overwrites *source.string* with the result. The second form returns the new string as the function's result, leaving *source.string* unchanged.

COS(*expr*)

Returns the cosine of *expr*. Angles are measured in degrees. If *expr* is a dynamic array, the function returns a similar dynamic array of results.

COUNT(*string*, *substring*)

Counts occurrences of *substring* within *string*.

COUNTS(*string*, *substring*)

Counts occurrences of *substring* within each element of dynamic array *string*, returning a similar dynamic array of results.

CREATE *file.var*
    {**ON ERROR** *statement(s)*}
    {**THEN** *statement(s)*}
    {**ELSE** *statement(s)*}
  Creates an empty directory file record after a previous **OPENSEQ** has reported that the record did not exist.

CREATE.FILE *path*   {**DIRECTORY** | **DYNAMIC**}
    {**GROUP.SIZE** *grpsz*}
    {**BIG.REC.SIZE** *bigrec*}
    {**MIN.MODULUS** *minmod*}
    {**SPLIT.LOAD** *split*}
    {**MERGE.LOAD** *merge*}
    {**VERSION** *ver*}
    {**ONERROR** *statement(s)*}
  Creates the operating system representation of a directory or dynamic hash file using the configuration information supplied via its optional parameters. Omitted parameters take their system defined default values. This statement does not create a corresponding VOC entry.

CREATE.SERVER.SOCKET(*addr***,** *port***,** *flags***)**
  Creates a server socket on which a program may wait for an incoming connection.

CROP(*string***)**
  Removes redundant mark characters from string.

CRT {*print.list*}
  Displays the *print.list* item(s) on the user's terminal. Where multiple items are specified as a comma separated lists, the display moves to the next tabulation column at each comma.

CSVDQ(*string* {**,** *delimiter*}**)**
  Dequotes a CSV string into a dynamic array.

CSV.MODE *mode* {**,***delim*}
  Sets quoting and delimiter rules for CSV output from FORMCSV(), PRINTCSV and WRITECSV.

DATA *expr*{**,** *expr...*}
  Inserts one or more lines of data into the data queue ready to be processed by subsequent **INPUT** statements.

DATE()
  Returns the internal representation of the day number of the current date.

DCOUNT(*string***,** *delimiter***)**
  Counts substrings delimited by delimiter within string.

DEBUG
  Enters debug mode for the program in which it was executed and all programs called by it.

DECRYPT(*data***,** *key***)**
  Decrypts data using the supplied key.

DEFFUN *name* {(*arg1* {,*arg2* ...})} {**CALLING "***subr***" | LOCAL**} {**VAR.ARGS**} {**KEY** *key*}
Defines a catalogued function that may be called from within the program. The optional
CALLING component allows the catalogue name of the function to be different from the name
of the function itself. The **LOCAL** keyword defines an internal function.

DEL *dyn.array<field* {, *value* {, *subvalue*}}>
Deletes the specified field, value or subvalue from the dynamic array.

DELETE(*dyn.array***,** *field* {, *value* {, *subvalue*}})
Returns a copy of *dyn.array* with the specified field, value or subvalue deleted.

DELETE *file.var*, *record.id* {**ON ERROR** *statement(s)*}
Deletes the specified record from the file and releases any lock held on this record. No error
occurs if the record does not exist.

DELETELIST *name*
Deletes the previously saved select list identified by *name* from the $SAVEDLISTS file. No
error occurs if the list does not exist.

DELETESEQ *file.name*, *id* {**ON ERROR** *statement(s)*}
        {**THEN** *statement(s)*} {**ELSE** *statement(s)*}
Deletes a record from a directory file without needing to open the file.

DELETESEQ *pathname* {**ON ERROR** *statement(s)*}
        {**THEN** *statement(s)*} {**ELSE** *statement(s)*}
Deletes an operating system file by pathname.

DELETEU *file.var*, *record.id* {**ON ERROR** *statement(s)*}
Deletes the specified record from the file, retaining any lock held on this record. No error occurs
if the record does not exist.

DIMENSION *mat*(*rows* {, *cols*})
DIM *mat*(*rows* {, *cols*})
Declares a matrix variable. Multiple matrices may be specified as a comma separated list.

DIR(*pathname***)**
Returns the contents of an operating system directory.

DISINHERIT *object*
Disinherits an object, removing it from the name search when locating a public variable,
function or subroutine.

DISPLAY {*print.list*}
Displays the print.list item(s) on the user's terminal. Where multiple items are specified as a
comma separated lists, the display moves to the next tabulation column at each comma.

DIV(*dividend***,** *divisor***)**
Returns the quotient from a division operation.

DOWNCASE(*string***)**
Returns the value of string with all letters converted to lower case.

DPARSE *string***,** *delimiter***,** *var1***,** *var2***,...**
    Splits delimited elements of string into variables *var1*, *var2*, etc.

DPARSE.CSV *string***,** *delimiter***,** *var1***,** *var2***,...**
    Splits delimited elements of string into variables *var1*, *var2*, etc, removing quotes in accordance
    with the CSV format standard.

DQUOTE(*expr***)**
    Returns a copy of its argument string enclosed in double quotes.

DQUOTES(*dyn.array***)**
    Returns a copy of the dynamic array argument with each element enclosed in double quotes.

DTX(*expr* **{,** *min.width***})**
    Converts the supplied *expr* value to hexadecimal. If the converted value is shorter than
    min.width, leading zeros are added.

EBCDIC(*expr***)**
    Returns the EBCDIC equivalent of the supplied ASCII string. The action of this function with
    non-ASCII characters is undefined.

ECHO OFF
ECHO ON
ECHO *expr*
    ECHO OFF will suppress keyboard echo until a subsequent **ECHO ON** statement. The
    ECHO *expr* format of this statement is equivalent to **ECHO ON** if the value of *expr* is
    non-zero and **ECHO OFF** if *expr* is zero.

ENCRYPT(*data***,** *key***)**
    Encrypts data using the supplied key.

END
    The **END** statement terminates a program, subroutine or a block of statements conditioned by
    the **THEN**, **ELSE**, **LOCKED** or **ON ERROR** keywords.

ENTER *name* **{(***arg.list***)}**   Without use of $MODE PICK.ENTER
ENTER @*var* **{(***arg.list***)}**
    Calls the named subroutine. In the second form, variable *var* contains the name of the
    subroutine.

ENTER *name*   With use of $MODE PICK.ENTER
ENTER @*var*
    Discards the current program and enters the named program. In the second form, variable *var*
    contains the name of the program.

ENV(*var.name***)**
    Retrieves the named operating system environment variable, returning its value. If the variable is
    not defined or *var.name* is invalid a null string is returned.

EPOCH()
    Returns the internal representation of the date and time as an epoch value.

EQS(*expr1*, *expr2*)
>   Compares corresponding elements of the dynamic arrays *expr1* and *expr2*, returning a similarly structured dynamic array of true / false values indicating the results of the comparison.

EQUATE *name* **LITERALLY "***string***"**
>   Assigns a name to the QMBasic elements in *string*. LITERALLY can be abbreviated to LIT.

EQUATE *name* **TO** *value*
>   Creates a symbolic name that can be used in place of the constant value.

EQUATE *name* **TO** *variable*
>   Creates a symbolic name that can be used in place of the variable name.

EQUATE *name* **TO** *matrix*(*index1* {**,** *index2*})
>   Assigns a name to a specific elements of a matrix. Declaration and dimensionality of the matrix are only checked when a reference to name is encountered during program compilation. The matrix index values *index1* and *index2* must be numbers. Reference to the zero element of a two dimensional matrix must include both index values.

EQUATE *name* **TO** *dyn.array*(*field* {**,** *value* {**,** *subvalue*}}>
>   Assigns a name to a specific elements of a dynamic array. The field, value and subvalue positions must be numeric constants.

EQUATE *name* **TO CHAR**(*seq*)
>   Assigns a name to a specific (usually non-printing) character.

All forms of **EQUATE** can be abbreviated to **EQU**. Multiple tokens may be equated on a single line by separating each definition by a comma.

ERRMSG *msg.id* {**,** *arg...*}
>   Displays the message with key msg.id from the ERRMSG file. The optional argument list is dependant on the message being displayed.
>
>   The ERRMSG file entry consists of one or more fields, each prefixed by an action code. The message is built up and displayed by processing each code in turn. The codes are:

A{*n*}	Display the next argument left aligned in a field of *n* characters. If *n* is omitted, the argument is displayed without any additional spaces.
B	Sound the terminal "bell".
D	Outputs the system date in the form dd mmm yyyy.
E	Outputs the *msg.id* enclosed in square brackets.
H*text*	Outputs the given *text*.
L{*n*}	Outputs *n* newlines. The value of *n* defaults to 1 if omitted.
R{*n*}	Display the next argument right aligned in a field of *n* characters. If *n* is omitted, the argument is displayed without any additional spaces.
S*n*	Displays *n* spaces.
T	Outputs the system time in the form hh:mm:ss.

>   The component parts of the message are output with no insertion of newlines except as explicitly specified in the ERRMSG entry.

EXECUTE *expr* **{TRAPPING ABORTS}**
    {**CAPTURING** *var*}
    {**PASSLIST** {*var*}}
    {**RTNLIST** *var*}
    {**SETTING** *status.var*}   or   {**RETURNING** *status.var*}
Executes the command defined by *expr*. The optional clauses are:

TRAPPING ABORTS	Abort events in the executed command cause return from the **EXECUTE** rather than aborting the program entirely.
CAPTURING	Captures command output in the named variable.
PASSLIST	Passes the named select list variable into the executed command as the default select list. The **PASSLIST** clause is ignored if var is omitted.
RTNLIST	Returns the default select list after execution of the command in the named select list variable.
STACKLIST	Saves the default list before executing the command and restores it on return to the program.
SETTING	Sets the value of @SYSTEM.RETURN.CODE into *status.var*.
RETURNING	Synonym for **SETTING**.

EXIT
Equivalent to a jump to the statement following the **REPEAT** or **NEXT** statement of the innermost loop structure.

EXP(*expr***)**
Returns the exponential of *expr*, that is, the mathematical constant e raised to the power *expr*. The value of e is about 2.71828. If *expr* is a numeric array (a dynamic array where all elements are numeric), the **EXP()** function operates on each element in turn and returns a numeric array with the same structure as *expr*.

EXTRACT(*dyn.array***,** *field* {**,** *value* {**,** *subvalue*}}**)**
Returns the specified field, value or subvalue from the dynamic array.

FCONTROL(*fvar***,** *action*{**,** *qualifier*}**)**
Performs a control action on an open file.

FIELD(*string***,** *delimiter***,** *occurrence* {**,** *count*}**)**
Extracts count substrings starting at substring *occurrence* from *string*. Substrings within *string* are delimited by the first character of *delimiter*. If *delimiter* is a null string, the entire *string* is returned. If the value of *occurrence* is greater than the number of delimited substrings in *string*, a null string is returned. If the value of *count* is greater than the number of delimited substrings in *string* starting at substring occurrence, the remainder of *string* is returned. Additional delimiters are not inserted.

FIELDS(*string***,** *delimiter***,** *occurrence* {**,** *count*}**)**
Similar to **FIELD()** but operates on a multi-valued string, returning a similarly structured dynamic array of results.

FIELDSTORE(*string***,** *delimiter***,** *i***,** *n***,** *rep.string***)**
Returns the result of storing *rep.string* as a delimited substring in *string*. The *string* variable is not changed. The action of **FIELDSTORE()** depends on the values of *i* and *n* and the number of substrings within *rep.string*.
If the value of the position expression, *i*, is less than one, a value of one is assumed. If there are fewer than *i* delimited substrings present in *string*, additional delimiters are added to reach the

required position.

If the value of the number of substrings expression, *n*, is positive, *n* substrings are replaced by the same number of substrings from *rep.string*. If *rep.string* contains fewer than *n* substrings, additional delimiters are inserted.

If the value of the number of substrings expression, *n*, is zero or negative, *n* substrings are deleted from *string* and the whole of *rep.string* is inserted regardless of the number of substrings that it contains.

FILE *name*, ...

Opens one or more files and allows access to data by field name.

FILE.EVENT(*path*, *event*)

Create a file event monitoring variable.

FILEINFO(*file.var*, *key*)

Returns information about an open file. The key value may be:

0	FL$OPEN	Check if file is open. Returns true (1) if file.var is associated with an open file, false (0) if it is not.
1	FL$VOCNAME	Returns the VOC name used to open the file.
2	FL$PATH	Returns pathname of open file.
3	FL$TYPE	File type. Returns one of
		FL$TYPE.DH (3) Dynamic file
		FL$TYPE.DIR (4) Directory file
		FL$TYPE.SEQ (5) Sequential file
		FL$TYPE.VFS (6)Virtual file system
5	FL$MODULUS	File modulus (dynamic files only)
6	FL$MINMOD	Minimum modulus (dynamic files only)
7	FL$GRPSIZE	Group size (dynamic files only)
8	FL$LARGEREC	Large record size (dynamic files only)
9	FL$MERGE	Merge load percentage (dynamic files only)
10	FL$SPLIT	Split load percentage (dynamic files only)
11	FL$LOAD	Current load percentage (dynamic files only)
13	FL$AK	File has alternate key indices (dynamic files only)
14	FL$LINE	Line to read or write next (sequential files only)
1000	FL$LOADBYTES	Current load bytes (dynamic files only)
1001	FL$READONLY	Returns true (1) is file is read-only
1002	FL$TRIGGER	Returns trigger function name, null if none
1003	FL$PHYSBYTES	Returns total size of file, excluding indices
1004	FL$VERSION	Internal file version number (dynamic files only)
1005	FL$STATS.QUERY	Returns true (1) if file statistics gathering is enabled
1006	FL$SEQPOS	File position (sequential files only)
1007	FL$TRG.MODES	Returns trigger mode flags
1008	FL$NOCASE	Returns true (1) if file uses case insensitive record ids.
1009	FL$FILENO	Returns the internal file number for *file.var*. This may change each time the file is opened.
1011	FL$AKPATH	Returns the pathname of the alternate key index directory. This is a null string if the indices are in their default location.
1012	FL$ID	Returns the id of the last record read from the file.
1013	FL$STATUS	Returns a dynamic array as for the STATUS statement.
1014	FL$MARK.MAPPING	Returns true if mark mapping is enabled, false if not (directory files).
1015	FL$RECORD.COUNT	Returns a count of the number of records in the file (dynamic files only).
1016	FL$PRI.BYTES	Physical size of the primary subfile in bytes (dynamic files

only). This figure will include space previously used by groups that have been discarded as the result of a merge operation.

1017	FL$OVF.BYTES	Physical size of the overflow subfile in bytes (dynamic files only). This figure will include space previously used by overflow blocks that are no longer active and are retained for future use.
1018	FL$NO.RESIZE	Is resizing inhibited on this file?
1019	FL$UPDATE	Update count, incremented by every write, delete or clear file.
1020	FL$ENCRYPTED	Returns true if file uses encryption.
1021	FL$WHO	Returns field mark delimited list of QM users numbers for users who have the file open. The process that executes the **FILEINFO()** function will not appear in this list if *file.var* is the only file variable referencing the file.
1022	FL$NETFILE	Returns true if this is a QMNet file, otherwise returns false.
1023	FL$SEQTYPE	Sequential file sub-type:

      FL$SEQTYPE.PORT (1) - Serial port
      FL$SEQTYPE.CHDEV (2) - Character device
      FL$SEQTYPE.FIFO (3) - FIFO
      FL$SEQTYPE.DRIVE (4) - Disk drive or other block device

1024	FL$REPLICATED	Returns a field mark delimited list of replication targets.
1025	FL$NOMAP	Returns true if this is a directory file with character translation of record ids suppressed.

FILELOCK *file.var* {**ON ERROR** *statement(s)*} {**LOCKED** *statement(s)*}
    {**THEN** *statement(s)*}
    {**ELSE** *statement(s)*}
Sets a file lock on the file open as *file.var*, ensuring exclusive access for update purposes.

FILEUNLOCK *file.var* {**ON ERROR** *statement(s)*}
    {**THEN** *statement(s)*}
    {**ELSE** *statement(s)*}
Releases a file lock, making the file available to other users. Read and update locks on records from the file are not affected.

FIND *string* **IN** *dyn.array* {**,** *occurrence*} **SETTING** *field*{**,** *value* {**,** *subvalue*}}
    {**THEN** *statement(s)*}
    {**ELSE** *statement(s)*}
Searches *dyn.array* for a field, value or subvalue equal to *string*. If found, the position of string within *dyn.array* is returned in the *field*, *value*, and *subvalue* variables and the **THEN** clause is executed. If not found or *dyn.array* is a null string, the *field*, *value*, and *subvalue* variables are unchanged and the **ELSE** clause is executed.

FIND *string* **IN** *dyn.array* {**,** *occurrence*} **SETTING** *field*{**,** *value* {**,** *subvalue*}}
    {**THEN** *statement(s)*}
    {**ELSE** *statement(s)*}
Searches *dyn.array* for a field, value or subvalue containing *string*. This need not be the entire field, value or subvalue. If found, the position of string within *dyn.array* is returned in the *field*, *value*, and *subvalue* variables and the **THEN** clause is executed. If not found or *dyn.array* is a null string, the *field*, *value*, and *subvalue* variables are unchanged and the **ELSE** clause is executed.

FLUSH *file.var*
>   {**THEN** *statement(s)*}
>   {**ELSE** *statement(s)*}
>   Flushes pending writes to a sequential file to disk.

FLUSH.DH.CACHE {**LOCAL** | **NO.WAIT**}
>   Flushes the dynamic file cache.

FMT(*expr*, *fmt.spec*)
>   Formats *expr* in a manner defined by *fmt.spec*.

FMTS(*expr*, *fmt.spec*)
>   Identical to **FMT**() except that it works on each element of dynamic array *expr* in turn, returning the result in a similarly delimited dynamic array.

FOLD(*string*, *width* {, *delim*})
>   Breaks *string* into sections delimited by *delim* (default field mark). Each section is at most *width* characters in length with the break from one section to the next occurring at a space where possible.

FOLDS(*string*, *width* {, *delim*})
>   Similar to **FOLD**() but works on each field, value or subvalue of *string* separately, returning a similarly structured dynamic array of folded strings

FOOTING {**ON** *print.unit*} *text*
>   Defines the text of a page footing and, optionally, control information determining the manner in which the text is output. A page footing is output whenever the bottom of the page is reached or on execution of a **PAGE** statement to terminate the current page.
>
>   The footing text may include the following control tokens enclosed in single quotes. Multiple tokens may appear within a single set of quotes.

> | | |
> |---|---|
> | **C** | Centres *text* on the line |
> | **D** | Insert the current date in the form mm/dd/yy |
> | **G** | Insert a gap. Spaces are inserted in the footing line at the position of each G control token such that the overall length of the line is the same as the printer unit width. A single use of the G token will right justify the subsequent text. Multiple G tokens will distribute spaces as evenly as possible. |
> | **H***n* | Sets horizontal position (column), numbered from 1. |
> | **L** | Start a new line |
> | **N** | Inhibit automatic display pagination |
> | **O** | Reverses the elements separated by G tokens in the current line on even numbered pages. This is of use when printing double sided reports. |
> | **P***n* | Inserts the page number, right justified in *n* spaces. If omitted, *n* defaults to 4. |
> | **S***n* | Inserts the page number, left justified in *n* spaces. If omitted, *n* defaults to 1. |
> | **T** | Insert current date and time in the form hh:mm:ss mm/dd/yy |

FOR *var* = *start.expr* **TO** *limit.expr* {**STEP** *step.expr*}
> *statement(s)*

NEXT {*var*}
>   Executes *statement(s)* for values of *var* from *start.expr* to *limit.expr* in increments of *step.expr*. If *step.expr* is positive, the loop continues while the value of *var* is less than or equal to *limit.expr*. If *step.expr* is negative, the loop continues while the value of *var* is greater than or equal to *limit.expr*. The value of *var* on leaving the loop is the last value for which the loop was

executed or *start.expr* if the initial value was already out of range. See also **CONTINUE**, **WHILE**, **UNTIL** and **EXIT**.

FOR *var* = *val1*, *val2*, *val3...*
  statement(s)
NEXT {*var*}
    Executes *statement(s)* for each of the values in the comma separated list. The value of *var* on leaving the loop is the last value for which the loop was executed. See also **CONTINUE**, **WHILE**, **UNTIL** and **EXIT**.

FOR EACH *var* **IN** *string*
  statement(s)
NEXT {*var*}
    Executes *statement(s)* for each of the mark character delimited items in *string*. The value of *var* on leaving the loop is the last value for which the loop was executed. See also **CONTINUE**, **WHILE**, **UNTIL** and **EXIT**.

FORMCSV(*string***)**
    Creates a CSV string from a dynamic array.

FORMLIST *dyn.array* {**TO** *list.no*}
    Clears any existing numbered select list *list.no* and creates a new list from the elements of *dyn.array*. This list may be separated by field marks, item marks or a mixture of both.

FORMLISTV *dyn.array* **TO** *var*
    Creates a select list variable from the elements of *dyn.array*. This list may be separated by field marks, item marks or a mixture of both.

FUNCTION *name*{(*arg1* {, *arg2...*}) {**VAR.ARGS**}}
    Defines a QMBasic program module as a function.

GES(*expr1***,** *expr2***)**
    Compares corresponding elements of the dynamic arrays *expr1* and *expr2*, returning a similarly structured dynamic array of true / false values indicating the results of the comparison.

GET *name*{(*args*)} {**VAR.ARGS**}
    Declares a public function in a class module (synonym for **PUBLIC FUNCTION**).

GET(ARG. {, *argno*) *var* {**THEN** *statement(s)*} {**ELSE** *statement(s)*}
    Retrieves command line arguments.

GET.MESSAGES()
    Returns any messages currently queued for display.

GET.PORT.PARAMS(*fvar***)**
    Returns a dynamic array containing the communications parameters for a serial port.

GETLIST *name* {**TO** *list.no*}
      {**THEN** *statement(s)*}
      {**ELSE** *statement(s)*}
    Restores the previously saved select list identified by name from the $SAVEDLISTS file. The disk copy is not removed by this operation.

GETNLS(*key***)**

Returns the value of the named national language support parameter. Available parameters are:

1	NLS$CURRENCY	Default currency symbol. Maximum 8 characters.
2	NLS$THOUSANDS	Default thousands separator character.
3	NLS$DECIMAL	Default decimal separator character.
4	NLS$IMPLCIIT.DECIMAL	Default decimal separator character for implicit conversions.

GETPU(*key***,** *unit***)**

Returns the print unit characteristic specified by key. Possible values of key are:

0	PU$DEFINED	Is print unit defined?
1	PU$MODE	Print unit mode
2	PU$WIDTH	Characters per line
3	PU$LENGTH	Lines per page
4	PU$TOPMARGIN	Top margin size
5	PU$BOTMARGIN	Bottom margin size
6	PU$LEFTMARGIN	Left margin size
7	PU$SPOOLFLAGS	Various print unit flags
9	PU$FORM	Form name (not used by all spoolers)
10	PU$BANNER	Banner page text
11	PU$LOCATION	Printer / file name
12	PU$COPIES	Number of copies to print
15	PU$PAGENUMBER	Current page number
1002	PU$LINESLEFT	Lines left on page
1003	PU$HEADERLINES	Lines occupied by header
1004	PU$FOOTERLINES	Lines occupied by footer
1005	PU$DATALINES	Lines between header and footer
1006	PU$OPTIONS	Options to be passed to the spooler
1007	PU$PREFIX	Pathname of prefix data file to be added to the start of the output
1008	PU$SPOOLER	Spooler to be used (ignored on Windows)
1009	PU$OVERLAY	Catalogued subroutine name for graphical page overlay
1010	PU$CPI	Characters per inch (may be non-integer value)
1011	PU$PAPER.SIZE	Paper size. See SYSCOM PCL.H
1012	PU$LPI	Lines per inch
1013	PU$WEIGHT	Font stroke weight. See SYSCOM PCL.H
1014	PU$SYMBOL.SET	Symbol set. See SYSCOM PCL.H
1015	PU$STYLE	Query processor style. See the Query processor STYLE option for details.
1016	PU$NEWLINE	Newline string for this print unit.
1017	PU$PRINTER.NAME	Name of printer.
1018	PU$FILE.NAME	File name for output directed to a file.
1019	PU$SEQNO	Hold file sequence number. See also @SEQNO.
1020	PU$FORM.QUEUE	Last assigned form queue number with SP.ASSIGN.
1021	PU$FORM.FEED	Form feed string for this print unit.
2000	PU$LINENO	Current line number

GETREM(*string***)**

Returns the offset of the remove pointer into string. The remove pointer is positioned on the character preceding the next fragment to be extracted. It is reset to zero when a new value is assigned to the string.

GOSUB *label*{**:**}

Enters the subroutine at the named label, saving the return address for a later **RETURN**.

GOSUB *label*{**:**}(*args*)

Enters the subroutine named label defined by a **LOCAL SUBROUTINE** statement, saving the return address for a later **RETURN**.

GOTO *label*{**:**}
GO {**TO**} *label*{**:**}

The program continues execution at the given label.

GTS(*expr1*, *expr2*)

Compares corresponding elements of the dynamic arrays *expr1* and *expr2*, returning a similarly structured dynamic array of true / false values indicating the results of the comparison.

HEADING { **NO.EJECT** } { **ON** *print.unit* } *text*

Defines the text of a page heading and, optionally, control information determining the manner in which the text is output. A page heading is output whenever the first line of output on a page is about to be printed or displayed. The **HEADING** statement normally causes subsequent output to appear on a new page. The **NO.EJECT** option defers the new heading until the start of the next page.

The heading text may include the following control tokens enclosed in single quotes. Multiple tokens may appear within a single set of quotes.

C	Centres *text* on the line
D	Insert the current date in the form mm/dd/yy
G	Insert a gap. Spaces are inserted in the footing line at the position of each G control token such that the overall length of the line is the same as the printer unit width. A single use of the G token will right justify the subsequent text. Multiple G tokens will distribute spaces as evenly as possible.
H*n*	Sets horizontal position (column), numbered from 1.
L	Start a new line
N	Inhibit automatic display pagination
O	Reverses the elements separated by G tokens in the current line on even numbered pages. This is of use when printing double sided reports.
P*n*	Inserts the page number, right justified in *n* spaces. If omitted, *n* defaults to 4.
S*n*	Inserts the page number, left justified in *n* spaces. If omitted, *n* defaults to 1.
T	Insert current date and time in the form hh:mm:ss mm/dd/yy

HUSH OFF {**SETTING** *var*}
HUSH ON {**SETTING** *var*}
HUSH *expr* {**SETTING** *var*}

The **HUSH ON** statement causes all output sent to the display by **CRT**, **DISPLAY** or **PRINT** statements to be suppressed. The **HUSH OFF** statement re-enables display. The **HUSH** *expr* format of this statement is equivalent to **HUSH ON** if the value of *expr* is non-zero and **HUSH OFF** if *expr* is zero.

The optional **SETTING** clause saves the previous state of display output control in *var* which can be used later to revert to that state. Alternatively, the previous state can be obtained using the **STATUS()** function immediately after the **HUSH** statement. In either case, the value is 1 if output was suppressed or 0 if it was enabled.

ICONV(*expr*, *conv.spec*)

Performs input conversion. Data in *expr* is converted from its external representation to the internal form.

ICONVS(*dynarray***,** *conv.spec***)**
　　Performs intput conversion. Data in each element of *dynarray* is converted from its external representation to the internal form to construct an equivalent dynamic array of converted values.

IDIV(*dividend***,** *divisor***)**
　　Returns the result of dividing integer value *dividend* by integer value *divisor*, rounded towards zero.

IF *expr* **THEN** *statement* {**ELSE** *statement*}
　　Executes one or other conditioned statements depending on the value of *expr*. The **ELSE** clause is optional. Either or both of the **THEN** and **ELSE** clauses may be moved to a new line.

IF *expr* **THEN**
　statement(s)
 {**END ELSE**
　statement(s)}
END
　　Executes one or other set of conditioned statements depending on the value of *expr*. The **ELSE** clause is optional.

　　A single **IF** statement may mix both the single line and multi-line formats.

IFS(*control.array***,** *true.array***,** *false.array***)**
　　Examines successive elements of *control.arr*ay and constructs a result array where elements are selected from the corresponding elements of either *true.array* or *false.array* depending on the *control.array* value.

IN *var* {**FOR** *timeout* {**THEN** *statement(s)*} {**ELSE** *statement(s)*}
　　Inputs the ASCII character number of a single character from the terminal with an optional timeout.

INDEX(*string***,** *substring***,** *occurrence***)**
　　Locates the specified *occurrence* of *substring* within *string* and returns its character position. If *occurrence* is less than one or the desired occurrence of *substring* is not found, the **INDEX()** function returns zero. If *substring* is null, the value of *occurrence* is returned.

INDEXS(*string***,** *substring***,** *occurrence***)**
　　Similar to **INDEX()** but *string* is a dynamic array. The function operates on each dynamic array element separately, locating the required occurrence of substring and returns a similarly structured dynamic array of results.

INDICES(*file.var***)**
　　Returns a field mark delimited list of alternate key index names for the file referenced via *file.var*.

INDICES(*file.var***,** *index.name***)**
　　Returns a dynamic array resembling a dictionary record for the index named by the *index.name* argument. This dynamic array has at least the following content. Other dictionary fields may also be present.
　　　　Field 1　　　　D or I dependant on the index type. Additionally, this field may be multivalued:

Value 2 Set to 1 if the index needs to be built, otherwise null
Value 3 Set to 1 if the index is null-suppressed (NO.NULLS option to
CREATE.INDEX), otherwise null
Value 4 Set to 1 if updates are enabled, otherwise null
Value 5 Internal AK number
Value 6 Sort mode of keys within an index entry
Value 8 Set to 1 if the index is case insensitive, otherwise null
Value 9 Set to 1 if the index is being built, otherwise null
Field 2 The field number (D-type) or the virtual attribute expression (I-type)
Field 5 L or R indicating whether the index is based on a left or right aligned sort
Field 6 S or M indicating whether the indexed data may be multi-valued
Field 16+ The compiled version of an I-type index

INHERIT *object*
Inherits an object, adding it from the name search when locating a public variable, function or
subroutine.

INMAT()
Returns information relating to the last use of the following statements:
DIMENSION 0 if successful, 1 if insufficient memory.
MATPARSE The number of elements assigned. Zero if overflows.
MATREAD The number of elements assigned. Zero if overflows.
MATREADL The number of elements assigned. Zero if overflows.
MATREADU The number of elements assigned. Zero if overflows.
OPEN The modulus of a dynamic file.

INMAT({*mat*}**)**
Returns the current dimensions of the matrix. If *mat* is a single dimensional matrix, **INMAT()**
returns the number of elements, excluding the zero element. If *mat* is a two dimensional matrix,
INMAT() returns the number of rows and columns as two values separated by a value mark.

INPUT *var* {**,** *length*} {**_**} {**:**} {**TIMEOUT** *wait*} {**HIDDEN**} {**NO.ECHO**} {**UPCASE**}
{**THEN** *statements*} { **ELSE** *statements*}
Input data from keyboard or **DATA** queue.

INPUT @(*col***,***row*) {**:**} *var* {**,** *length*} {**_**} {**:**} {**APPEND**} {**EDIT**} {**HIDDEN**} {**NO.ECHO**}
{**OVERLAY**} {**PANNING**} {**TIMEOUT** *wait*} {**UPCASE**}
{**THEN** *statements*} { **ELSE** *statements*}
Input data from keyboard or **DATA** queue at the specified screen position.

INPUTCLEAR
Any type-ahead data is cleared. Data stored by the **DATA** statement is not affected.

INPUTCSV *var1***,** *var2***,** ...
Input CSV format data from keyboard or **DATA** queue.

INPUTERR *expr*
Displays an error message which is removed from the screen when the next input is entered.

INPUTFIELD @(*x***,** *y*) {**,**} {**:**} *var***,** length {**_**} {**:**} {*format*} {**APPEND**} {**EDIT**} {**HIDDEN**}
{**NO.ECHO**} {**OVERLAY**} {**PANNING**} {**TIMEOUT** *wait*} {**UPCASE**}
{**THEN** *statements*} { **ELSE** *statements*}
Input data from keyboard or **DATA** queue with special handling of control codes.

INS *string* **BEFORE** *dyn.array*<*field* {**,** value {**,** *subvalue*}}**>**
INSERT(*dyn.array***,** *field* {**,** *value* {**,** *subvalue*}} **,** *string*)
    Inserts a field, value or subvalue into a dynamic array.

INT(*expr*)
    Returns the integer value of *expr*, rounded towards zero.

ITYPE(*itype*)
    Executes the compiled I-type expression in the supplied dictionary style record. @RECORD and @ID are used as source data for the calculation.

KEYCODE({*timeout*})
    Reads a single keystroke from the keyboard, using the terminfo database to recognise standard control codes.

KEYEDIT (*action***,** *key*)**,** (*action***,** *key*)**,** ...
    Defines editing keys for use with **INPUT** @.

KEYEXIT (*action***,** *key*)**,** (*action***,** *key*)**,** ...
    Defines exit keys for use with **INPUT** @.

KEYIN({*timeout*})
    Reads a single keystroke from the keyboard.

KEYINC({*timeout*})
    Reads a single keystroke from the keyboard, honouring case inversion.

KEYINR({*timeout*})
    Reads a single keystroke from the keyboard in raw mode (no internal processing).

KEYREADY()
    Tests for data entered at the keyboard.

KEYTRAP (*action***,** *key*)**,** (*action***,** *key*)**,** ...
    Defines trap keys for use with **INPUT** @.

LEN(*string*)
    Returns the length of a string.

LENS(*string*)
    Returns the length of a length of each element of a dynamic array.

LES(*expr1***,** *expr2*)
    Compares corresponding elements of the dynamic arrays *expr1* and *expr2*, returning a similarly structured dynamic array of true / false values indicating the results of the comparison.

LISTINDEX(*list***,** *delimiter***,** *item*)
    Returns the position of item in a *list* separated by *delimiter*.

LN(*expr*)
    Returns the natural log of a value.

LOCAL FUNCTION *name*{(*args*)}
  Declares an internal function that may have private local variables.

LOCAL SUBROUTINE *name*{(*args*)}
  Declares an internal subroutine that may have private local variables.

LOCATE *string* **IN** *dyn.array*<*field* {**,***value* {**,** *subvalue*}}> {**BY** *order*} {**SETTING** *var*}
    {**THEN** *statement(s)*}
    {**ELSE** *statement(s)*}
  Searches a dynamic array for a given field, value or subvalue (without $MODE UV.LOCATE).

LOCATE *string* **IN** *dyn.array*<*field* {**,***value* }> {**,** *start*}{**BY** *order*} {**SETTING** *var*}
    {**THEN** *statement(s)*}
    {**ELSE** *statement(s)*}
  Searches a dynamic array for a given field, value or subvalue (with $MODE UV.LOCATE).

LOCATE(*string***,** *dyn.array*{**,***field* {**,***value* }}**;** *var* {**;** *order*})   [Used as a statement]
    {**THEN** *statement(s)*}
    {**ELSE** *statement(s)*}
  Searches a dynamic array for a given field, value or subvalue (Pick style syntax).

LOCATE(*string***,** *dyn.array***,** *field* {**,***value* {**,** *subvalue*}} {**;** *order*})   [Used as a function]
  Returns position of a given field, value or subvalue in a dynamic array.

LOCK *lock.num* {**THEN** *statement(s)*} {**ELSE** *statement(s)*}
  Obtains one of 64 system wide task locks.

LOGMSG *text*
  Add a message to the system error log.

LOOP
  statement(s)
REPEAT
  Repeats the enclosed statements until the loop is terminated. See also **CONTINUE**, **WHILE**,
  UNTIL and **EXIT**.

LOWER(*string*)
  Converts mark characters in a string to the next lower level mark.

LTS(*expr1***,** *expr2*)
  Compares corresponding elements of the dynamic arrays *expr1* and *expr2*, returning a similarly
  structured dynamic array of true / false values indicating the results of the comparison.

MARK.MAPPING *file.var***, OFF**
MARK.MAPPING *file.var***, ON**
MARK.MAPPING *file.var***,** *expr*
  Determines how field marks are handled when reading or writing from a directory file.

MAT *matrix* = *expr*
  Assigns a value to all elements of a matrix.

MAT *matrix* = **MAT** *src.matrix*
  Copies one matrix to another.

MATBUILD *var* **FROM** *mat* {**,** *start.expr* {**,** *end.expr*} {**USING** *delimiter*}
Constructs a dynamic array from the elements of a matrix.

MATCHESS(*dyn.arr***,** *pattern*)
Matches each element of a dynamic array against a pattern, returning an equivalently structured dynamic array of true/false values.

MATCHFIELD(*string***,** *pattern***,** *element*)
Extracts a portion of a string that matches a pattern element.

MATPARSE *mat* **FROM** *string* {**,** *delimiter*}
MATPARSE *mat* **FROM** *string* {**USING** *delimiter*}
Breaks a delimited string into component substrings, assigning each to an element of a matrix.

MATREAD{**L|U**} *mat* **FROM** *file.var***,** *record.id* {**ON ERROR** *statement(s)*}
　　　{**LOCKED** *statement(s)*}
　　　{**THEN** *statement(s)*}
　　　{**ELSE** *statement(s)*}
Reads a record from a file, assigning each field to an element of a matrix. The **MATREADL** statement sets a read lock on the record. The **MATREADU** statement sets an update lock on the record.

MATREADCSV *matrix* **FROM** *file.var*
　　　{**THEN** *statement(s)*}
　　　{**ELSE** *statement(s)*}
Reads CSV format data from a directory file record previously opened for sequential access.

MATWRITE{**U**} *mat* **TO** *file.var***,** *record.id* {**ON ERROR** *statement(s)*}
Builds a record from successive elements of a matrix and writes this to a file. The **MATWRITEU** statement preserves any lock on the record being written.

MAX (*a***,** *b*)
Returns the greater of two values.

MAXIMUM(*dyn.array*)
Returns the greatest numeric value in a dynamic array.

MD5 (*string*)
Returns the 32 digit MD5 message digest value for the given string.

MIN (*a***,** *b*)
Returns the lesser of two values.

MINIMUM(*dyn.array*)
Returns the lowest numeric value in a dynamic array.

MOD(*dividend***,** *divisor*)
Returns the modulus value of one value divided by another.

MODS(*dividend***,** *divisor*)
Returns a dynamic array of the modulus values for division of corresponding elements of two dynamic arrays.

MVDATE(*epoch***)**
Returns the multivalue style date from an epoch value.

MVTIME(*epoch***)**
Returns the multivalue style time from an epoch value.

MVEPOCH(*time.string***)**
MVEPOCH(*date***,** *time***)**
Returns the epoch value for a given multivalue style date and time.

MVDATE.TIME(*epoch***)**
Returns the multivalue style date and time from an epoch value.

NAP *time*
Causes the program to pause for a given number of milliseconds.

NEG(*expr***)**
Returns the arithmetic inverse of a value.

NEGS(*dynarray***)**
Returns the arithmetic inverse of each element of a dynamic array.

NES(*expr1***,** *expr2***)**
Compares corresponding elements of the dynamic arrays *expr1* and *expr2*, returning a similarly structured dynamic array of true / false values indicating the results of the comparison.

NOBUF *file.var*
{**THEN** *statement(s)*}
{**ELSE** *statement(s)*}
Turns off buffering for a record opened using **OPENSEQ**.

NOT(*expr)*
Returns the logical inverse expr.

NOTS(*dynarray***)**
Returns the logical inverse of each element of a dynamic array.

NULL
Performs no action.

NUM(*string***)**
Tests whether a string can be converted to a number.

NUMS(*dynarray***)**
Tests whether each element of a dynamic array can be converted to a number.

OBJECT(*cat.name* {**,** *args*}**)**
Instantiates an object defined by the class module catalogued as cat.name.

OBJINFO(*var* **,** *key***)**
Returns information about an object variable.
    0   OI$ISOBJ   Is var an object variable?

1 OI$CLASS   Class module catalogue name

OCONV(*expr***,** *conv.spec***)**
   Performs output conversion. Data in *expr* is converted from its internal representation to the external form.

OCONVS(*dynarray***,** *conv.spec***)**
   Performs output conversion. Data in each element of dynarray is converted from its internal representation to the external form to construct an equivalent dynamic array of converted values.

ON *expr* **GOSUB** *label1*{**:**}**,** *label2*{**:**}**,** *label3*{**:**}
   Enters one of a list of internal subroutines depending on the value of *expr*.

ON *expr* **GOTO** *label1*{**:**}**,** *label2*{**:**}**,** *label3*{**:**}
ON *expr* **GO** {**TO**} *label1*{**:**}**,** *label2*{**:**}**,** *label3*{**:**}
   Jumps to one of a list of labels depending on the value of *expr*.

OPEN {*dict.expr***,**} *filename.expr* {**NON.TRANSACTIONAL**} {**READONLY**} {**SYNC**} **TO** *file.var* {**ON ERROR** *statement(s)*}
      {**THEN** *statement(s)*}
      {**ELSE** *statement(s)*}
   Opens a directory file or dynamic file, associating it with a file variable.

OPENPATH *pathname* {**NON.TRANSACTIONAL**}{**READONLY**} {**SYNC**} **TO** *file.var* {
ON ERROR *statement(s)*}
      {**THEN** *statement(s)*}
      {**ELSE** *statement(s)*}
   Opens a directory file or dynamic file by pathname, associating it with a file variable.

OPENSEQ *file.name***,** *id* {**APPEND** | **NOBUF** | **NO.MAP** | **OVERWRITE** | **READONLY**} **TO** *file.var*
      {**ON ERROR** *statement(s)*}
      {**LOCKED** *statement(s)*}
      {**THEN** *statement(s)*}
      {**ELSE** *statement(s)*}
   Opens a record of a directory file for sequential access.

OPENSEQ *pathname* {**APPEND** | **NOBUF** | **NO.MAP** | **OVERWRITE** | **READONLY**} **TO** *file.var*
      {**ON ERROR** *statement(s)*}
      {**LOCKED** *statement(s)*}
      {**THEN** *statement(s)*}
      {**ELSE** *statement(s)*}
   Opens a file, a device or a pipe by pathname for sequential access.

OPEN.SOCKET(*addr***,** *port***,** *flags***)**
   Opens a data socket for an outgoing connection.

ORS(*expr1***,** *expr2***)**
   Constructs the logical or of corresponding elements of two dynamic arrays, returning a similarly structured dynamic array of true / false values.

OS.ERROR()
Returns operating system error information.

OS.EXECUTE *expr* {**CAPTURING** *var*}
Executes *expr* as an operating system command.

OSDELETE *path*
Deletes an operating system file by pathname.

OSREAD *var* **FROM** *path* {**ON ERROR** *statement(s)*}
    {**THEN** *statement(s)*}
    {**ELSE** *statement(s)*}
Reads an operating system file by pathname.

OSWRITE *var* **TO** *path* {**ON ERROR** *statement(s)*}
Writes an operating system file by pathname.

OUTERJOIN({**DICT**} *file.name*, *field.name*, *value*)
Returns a list of record ids for records where the named field holds a given value.

PAGE {**ON** *print.unit*} {*page.no*}
Advances a print unit to a new page.

PAUSE {*timeout*}
Suspend execution until awoken by another process.

PRECISION *expr*
Sets the maximum number of decimal places to appear when converting numeric values to strings.

PRINT {**ON** *print.unit*} {*print.list*}
Outputs data to a print unit.

PRINTCSV {**ON** *print.unit*} *var1*, *var2*, ... {**:**}
Outputs CSV format data to a print unit.

PRINTER CLOSE {**ON** *print.unit*}
Closes one or all print units.

PRINTER DISPLAY {**ON** *print.unit*}
    {**ON ERROR** *statement(s)*}
    {**THEN** statement(s)}
    {**ELSE** *statement(s)*}
Directs output sent to a print unit to the display.

PRINTER FILE {**ON** *print.unit*} *file.name*, *record.name*
    {**ON ERROR** *statement(s)*}
    {**THEN** *statement(s)*}
    {**ELSE** *statement(s)*}
Associates a file with a print unit.

PRINTER NAME {**ON** *print.unit*} *printer.name*
    {**ON ERROR** *statement(s)*}
    {**THEN** *statement(s)*}
    {**ELSE** *statement(s)*}
  Associates a named printer device with a print unit.

PRINTER ON
PRINTER OFF
  Determine whether output from **PRINT** statements to the default print unit (unit 0) is directed to the display or to the printer.

PRINTER RESET
  Resets the default print unit and display output.

PRINTERR *expr*
  Displays an error message which is removed from the screen when the next input is entered.

PRIVATE *var*, *...*
  Declares private variables in a local subroutine or a class module.

PROCREAD *var* {**THEN** *statement(s)*} {**ELSE** *statement(s)*}
  Reads data from the PROC primary input buffer.

PROCWRITE *expr*
  Writes data to the PROC primary input buffer.

PROGRAM *name*
  Introduces a program.

PROMPT *expr*
  Sets the character to be used as the prompt in **INPUT** statements.

PTERM(*key*, *value*)
  Sets, clears or queries a terminal setting. Possible values of *key* are:

1	PT$BREAK	Interpret break character as break key? (boolean)
2	PT$INVERT	Invert case of alphabetic characters on input? (boolean)
3	PT$BRKCH	Sets break character to be char(*value*). The *value* must be in the range 1 to 31. For any other value, the current setting is returned.
4	PT$PAGE.PAUSE	Set/clear page pause (boolean)
5	PT$MARK	Set/clear translation of characters 28-30 on keyboard input to field, value and subvalue marks. (boolean)

PUBLIC *var* {**READONLY**}, *...*
  Declares public property variables in a class module.

PUBLIC FUNCTION *name*{(*args*)} {**VAR.ARGS**}
  Declares a public function in a class module.

PUBLIC SUBROUTINE *name*{(*args*)} {**VAR.ARGS**}
  Declares a public function in a class module.

PWR(*expr***,** *pwr.expr***)**
> Returns the value of a number raised to a given power.

QUOTE(*expr***)**
> Returns a copy of its argument string enclosed in double quotes.

QUOTES(*dyn.array***)**
> Returns a copy of the dynamic array argument with each element enclosed in double quotes.

RAISE(*string***)**
> Converts mark characters in a string to the next higher level mark.

RANDOMIZE *expr*
> Initialises the random number generator.

RDIV(*dividend***,** *divisor***)**
> Returns the rounded integer result of dividing two values

READ *var* **FROM** *file.var***,** *record.id* {**ON ERROR** *statement(s)*}
> {**THEN** *statement(s)*}
> {**ELSE** *statement(s)*}
> Reads a record from a previously opened file.

READ.SOCKET(*skt***,** *max.len***,** *flags***,** *timeout***)**
> Reads data from a socket.

READBLK *var* **FROM** *file.var***,** *bytes*
> {**ON ERROR** *statement(s)*}
> {**THEN** *statement(s)*}
> {**ELSE** *statement(s)*}
> Reads a given number of bytes from the current file position in a record previously opened using **OPENSEQ**.

READCSV FROM *file.var* **TO** *var1***,** *var2***,** *...*
> {**THEN** *statement(s)*}
> {**ELSE** *statement(s)*}
> Reads CSV format data from a directory file record previously opened for sequential access.

READL *var* **FROM** *file.var***,** *record.id* {**ON ERROR** *statement(s)*}
> {**LOCKED** *statement(s)*}
> {**THEN** *statement(s)*}
> {**ELSE** *statement(s)*}
> Reads a record from a previously opened file, setting a shareable read lock on the record.

READLIST *var* {**FROM** *list.no*}
> {**THEN** *statement(s)*}
> {**ELSE** *statement(s)*}
> Reads a select list into a dynamic array.

READNEXT *var* {**FROM** *list.no*}
　　{**ON ERROR** statement(s)}
　　{**THEN** *statement(s)*}
　　{**ELSE** *statement(s)*}
　Returns the next item from an active select list.

READNEXT *var***,** *valpos***,** *subvalpos* {**FROM** *list.no*}
　　{**ON ERROR** *statement(s)*}
　　{**THEN** *statement(s)*}
　　{**ELSE** *statement(s)*}
　Returns the next item and positional information from an active exploded select list.

READSEQ *var* **FROM** *file.var* {**ON ERROR** *statement(s)*}
　　{**THEN** *statement(s)*}
　　{**ELSE** *statement(s)*}
　Reads the next field from a directory file record previously opened for sequential access.

READU *var* **FROM** *file.var***,** *record.id* {**ON ERROR** *statement(s)*}
　　{**LOCKED** *statement(s)*}
　　{**THEN** *statement(s)*}
　　{**ELSE** *statement(s)*}
　Reads a record from a previously opened file, setting an update lock on the record.

READV *var* **FROM** *file.var***,** *record.id***,** *field.expr*
　　{**ON ERROR** *statement(s)*}
　　{**THEN** *statement(s)*}
　　{**ELSE** *statement(s)*}
　Reads a specific field from a record of a previously opened file.

READVL *var* **FROM** *file.var***,** *record.id***,** *field.expr*
　　{**ON ERROR** *statement(s)*}
　　{**LOCKED** *statement(s)*}
　　{**THEN** *statement(s)*}
　　{**ELSE** *statement(s)*}
　Reads a specific field from a record of a previously opened file, setting a shareable read lock on the record.

READVU *var* **FROM** *file.var***,** *record.id***,** *field.expr*
　　{**ON ERROR** *statement(s)*}
　　{**LOCKED** *statement(s)*}
　　{**THEN** *statement(s)*}
　　{**ELSE** *statement(s)*}
　Reads a specific field from a record of a previously opened file, setting an update lock on the record.

RECORDLOCKED(*file.var***,** *record.id***)**
　The RECORDLOCKED() function indicates whether a given record is locked. Return values are:

Value	Token	Lock state
-3	LOCK$OTHER.FILELOCK	Another user holds a file lock
-2	LOCK$OTHER.READU	Another user holds an update lock
-1	LOCK$OTHER.READL	Another user holds a read lock
0	LOCK$NO.LOCK	The record is not locked

1	LOCK$MY.READL	This user holds a read lock
2	LOCK$MY.READU	This user holds an update lock
3	LOCK$MY.FILELOCK	This user holds a file lock

RECORDLOCKL *file.var***,** *record.id* {**ON ERROR** *statement(s)*}
{**LOCKED** *statement(s)*}
Sets a shareable read lock on a record.

RECORDLOCKU *file.var***,** *record.id* {**ON ERROR** *statement(s)*}
{**LOCKED** *statement(s)*}
Sets an update lock on a record.

RELEASE {*file.var*{**,** *record.id*}} {**ON ERROR** *statement(s)*}
Releases read, update or file locks.

REM(*dividend***,** *divisor***)**
Returns the remainder when one value is divided by another.

REMARK *text*
REM *text*
Enters comment text into a program.

REMOVE *string* **FROM** *dyn.array* **SETTING** *var*
REMOVE(*dyn.array***,** *var***)**
Extracts characters from a dynamic array up to the next mark character.

REMOVEF(string{**,** *delimiter*{**,** *count*}}**)**
Extracts data from a delimited string.

REMOVE.BREAK.HANDLER
Deactivates a break handler previously established to be called on use of the break key.

REPLACE(*dyn.array***,** *field* {**,** *value* {**,** *subvalue*}} **,** *string***)**
Replaces a field, value or subvalue of a dynamic array, returning the result.

RESTORE.SCREEN *image***,** *restore.state*
Restores a rectangular portion of the display screen image previously saved using
SAVE.SCREEN().

RETURN
Returns from a local subroutine entered by **GOSUB** or an external subroutine entered by **CALL**
.

RETURN *expr*
Returns from a function, passing *expr* to the caller as the value of the function.

RETURN FROM PROGRAM
Returns from a **CALL**, regardless of internal subroutine depth.

RETURN TO *label*{**:**}
Exits from a local subroutine entered by **GOSUB**, discarding the return address and jumping to
label. If not in a local subroutine, this statement returns from the program.

REUSE(*num.array***)**
Determines how arithmetic operators applied to numeric arrays handle unequal numbers of fields, values or subvalues.

RND(*expr***)**
Returns a random integer within a range defined by *expr*.

ROUNDDOWN(*value***,** *increment***)**
Returns *value* rounded towards zero in steps of *increment*.

ROUNDUP(*value***,** *increment***)**
Returns *value* rounded away from zero in steps of *increment*.

ROLLBACK
Terminates a transaction, discarding all cached updates.

RQM {*time*}
Causes the program to pause for a given number of seconds or until a specific time.

RTRANS({**DICT**} *file.name***,** *record.id***,** *field***,** *action***)**
Returns a field or the entire record from a named data file. Normally only used in dictionary I-type items. **RTRANS()** differs from **TRANS()** in that it does not lower mark characters in the returned data.

SAVE.SCREEN(*col***,** *line***,** *width***,** *height***)**
Saves a rectangular portion of the display screen image. Only available on some terminal types.

SAVELIST *name* {**FROM** *list.no*}
　　{**THEN** *statement(s)*}
　　{**ELSE** *statement(s)*}
Saves an active select list to the $SAVEDLISTS file.

SEEK *file.var* {**,** *offset*{**,** *relto* }}
　　{**THEN** *statement(s)*}
　　{**ELSE** *statement(s)*}
Sets the current read / write position in a directory file record previously opened for sequential access.

SELECT *file.var* {**TO** *list.no*} {**ON ERROR** *statement(s)*}
Creates a select list containing all record keys from a file. The type of list is dependent on compiler mode settings.

SELECT.SOCKET(*skt.array***,** *timeout***)**
Monitor events on multiple sockets.

SELECTE TO *list.var*
Transfers the default select list to a select list variable.

SELECTN *file.var* {**TO** *list.no*} {**ON ERROR** *statement(s)*}
Creates a numbered select list containing all record keys from a file.

SELECTV *file.var* **TO** *list.no* {**ON ERROR** *statement(s)*}
Creates a select list variable containing all record keys from a file.

SELECTINDEX *index.name* {**,** *value*} **FROM** *file.var* {**TO** *list.no*}
Creates a select list from an alternate key index entry.

SELECTINFO(*list.no***,** *key*)
SELECTINFO(*var***,** *key*)
Returns information about a select list.

SELECTLEFT *index.name* **FROM** *file.var* {**SETTING** *key*} {**TO** *list.no*}
SELECTRIGHT *index.name* **FROM** *file.var* {**SETTING** *key*} {**TO** *list.no*}
Create a select list from the entry in an alternate key index to the left or right of the last entry processed.

SENTENCE()
Returns the command line that started the current program (Same as @SENTENCE).

SEQ(*char*)
Returns the ASCII character set position value of a character.

SERVER.ADDR(*server.name*)
Returns the IP address for the given server name.

SET *name*{(*args*)} {**VAR.ARGS**}
Declares a public subroutine in a class module (synonym for **PUBLIC SUBROUTINE**).

SET.BREAK.HANDLER *handler*
Establishes a break handler to be called on use of the break key.

SET.EXIT.STATUS *value*
Sets the final exit status returned by QM to the operating system.

SET.PORT.PARAMS(*fvar***,** *params*)
Sets the communications parameters for a serial port.

SET.SOCKET.MODE(*skt***,** *key***,** *value*)
Sets a mode parameter for an open socket.

SET.TIMEZONE *zone*
Sets time zone for use by the epoch conversion code.

SETLEFT *index.name* **FROM** *file.var*
SETRIGHT *index.name* **FROM** *file.var*
Set the scanning position of an alternate key index at the extreme left or right of the data.

SETNLS *key***,** *value*
Sets the value of a national language support parameter.

Parameter	Key	Meaning
1	NLS$CURRENCY	Default currency symbol. Maximum 8 characters.
2	NLS$THOUSANDS	Default thousands separator character.
3	NLS$DECIMAL	Default decimal separator character.
4	NLS$IMPLCIIT.DECIMAL	Default decimal separator character for implicit conversions.

SETPU *key***,** *unit***,** *value*

Sets the characteristics of a print unit.

Key	Token	Meaning
1	PU$MODE	Print unit mode
2	PU$WIDTH	Characters per line
3	PU$LENGTH	Lines per page
4	PU$TOPMARGIN	Top margin size
5	PU$BOTMARGIN	Bottom margin size
6	PU$LEFTMARGIN	Left margin size
7	PU$SPOOLFLAGS	Various print unit flags
9	PU$FORM	Form name (not used by all spoolers)
10	PU$BANNER	Banner page text
11	PU$LOCATION	Printer / file name
12	PU$COPIES	Number of copies to print
15	PU$PAGENUMBER	Current page number
1006	PU$OPTIONS	Options to be passed to the spooler
1007	PU$PREFIX	Pathname of prefix data file to be added to the start of the output
1008	PU$SPOOLER	Spooler to be used (ignored on Windows)
1009	PU$OVERLAY	Catalogued subroutine name for graphical page overlay
1010	PU$CPI	Characters per inch (may be non-integer value)
1011	PU$PAPER.SIZE	See SYSCOM KEYS.H for size tokens
1012	PU$LPI	Lines per inch. Must be 1, 2, 3, 4, 6, 8, 12, 16, 24, 48
1013	PU$WEIGHT	Font stroke weight. See SYSCOM PCL.H
1014	PU$SYMBOL.SET	Symbol set. See SYSCOM PCL.H
1015	PU$STYLE	Query processor style. See the Query processor STYLE option.
1016	PU$NEWLINE	Newline string for this print unit.
1017	PU$PRINTER.NAME	Name of printer.
1018	PU$FILE.NAME	File name for output directed to a file.
1021	PU$FORM.FEED	Form feed string for this print unit.

SETREM *offset* **ON** *string*

Sets the remove pointer of a string.

SHIFT(*value***,** *shift.len*)

Performs a logical right bit-shift of *shift.len* places on integer value. A negative *shift.len* shifts left.

SIN(*expr*)

Returns the sine of a value.

SLEEP {*time*}

Causes the program to pause for a given number of seconds or until a specific time.

SOCKET.INFO(*skt***,** *key*)

Returns information about an open socket.

SORT.COMPARE(*string1***,** *string2***,** *mode***,** *no.case*)

Compare items according to sort rules.

SOUNDEX(*string*)

Returns a four character string determined by the phonetic content of a string,

SOUNDEXS(*string*)
Constructs a dynamic array in which each element consists of the phonetic Soundex code for the content of the corresponding element of dynarray,

SPACE(*count*)
Returns a string consisting of a given number of spaces.

SPACES(*dynarray*)
Constructs a dynamic array in which each element consists of the number of spaces specified by the corresponding element of *dynarray*.

SPLICE(*array1*, *string*, *array2*)
Returns the result of concatenating corresponding dynamic array components (fields, values and subvalues) with string inserted between each element.

SQRT(*expr*)
Returns the square root of a value.

SQUOTE(*expr*)
Returns a copy of its argument string enclosed in single quotes.

SQUOTES(*dyn.array*)
Returns a copy of the dynamic array argument with each element enclosed in single quotes.

SSELECT *file.var* {**TO** *list.no*} {**ON ERROR** *statement(s)*}
Creates a select list containing all record keys from a file sorted into left justified ascending order.

STATUS()
Returns information following execution of certain other statements. In many cases, this information gives details of an error condition.

STATUS *var* **FROM** *file.var* **THEN** *statement(s)* **ELSE** *statement(s)*
Returns a dynamic array of information about an open file.

STOP {*expr*}
Terminates the current program. The interpretation of *expr* depends on the setting of the PICK.ERRMSG option.

STOPE {*expr*}
Terminates the current program using Pick style message handling.

STOPM {*print.list*}
Terminates the current program, displaying *print.list*.

STR(*string*, *count*)
Returns a string made up of a given number of repeated occurrences of another string.

STRS(*dynarray*, *count*)
Returns a dynamic array made up of count occurrences of each element of *dynarray*.

SUBR(*name* {**,***arg1* {**,** *arg2*...}**)**
Calls a subroutine as a function in an expression. It is normally only used in dictionary I-type items.

SUBROUTINE *name*{(*arg1* {**,** *arg2*...}) {**VAR.ARGS**}}
Introduces a subroutine.

SUBSTITUTE(*dyn.array***,** *old.list***,** *new.list* {**,** *delimiter*}**)**
Performs substring replacement on successive elements of a dynamic array, returning a similarly structured dynamic array of results.

SUBSTRINGS(*dyn.array***,** *start***,** *length***)**
Performs substring extraction on successive elements of a dynamic array, returning a similarly structured dynamic array of results.

SUM(*expr***)**
Eliminates the lowest level of a dynamic array by adding the elements to form an item of the next highest level.

SUMMATION(*expr***)**
Returns the total value of all elements of a numeric array.

SWAP(*string***,** *old***,** *new*{**,** *occurrence*{**,** *start*}}**)**
Replaces the specified occurrences of *old* within *string* by *new*. *Occurrence* evaluates to the number of occurrences of *old* to be replaced. If omitted or specified as a value of less than one, all occurrences are replaced. *Start* specifies the first occurrence to be replaced. If omitted or specified as a value of less than one it defaults to one.

SWAPCASE(*string***)**
Inverts the case of all alphabetic characters in a string.

SYSTEM(*key***)**
Returns information regarding the status of various aspects of the system.

Key	Function
1	Returns 1 if a PRINTER ON statement is in effect
2	Current page width of the default print unit
3	Current page length of the default print unit
4	Lines remaining on current page of the default print unit
5	Current page number of the default print unit
6	Current line number of the default print unit
7	Terminal type (same as @TERM.TYPE)
9	Cumulative processor time used (mS) by the QM session
10	Input waiting in the DATA queue? (1 if so, 0 if not)
11	Select list 0 active? (1 if so, 0 if not)
12	Time in seconds since midnight (same as TIME())
18	User number (same as @USERNO)
19	Returns a unique value formed from the internal form date and time as two five digit numbers. If the function is used more than once system wide in the same second, an alphabetic suffix is added to create a unique value.
23	Break key enabled? (1 if so, 0 if not)
24	Input echo enabled? (1 if so, 0 if not)
25	Is this a phantom process?
26	Returns the current input prompt character
27	Returns the operating system uid for the user's process. (Not Windows or PDA)

28	Returns the operating system effective uid for the user's process. (Not Windows or PDA)
29	Returns the operating system gid for the user's process. (Not Windows or PDA)
30	Returns the operating system effective gid for the user's process. (Not Windows or PDA)
31	Licence number
32	Returns the system directory pathname
38	Returns temporary directory pathname
42	Returns telnet connection IP address, null for a console user
91	Returns 1 on Windows, 0 on Linux or FreeBSD
1000	Returns 1 if EXECUTE CAPTURING is in effect, 0 otherwise
1001	Returns 1 if case inversion is enabled, 0 otherwise
1002	Returns the program call history. This is a dynamic array in which each program is represented by a field, the current program being in field 1. The first value in each field contains the program name. Subsequent values are divided into two subvalues containing the program address and line number (where available) for each internal subroutine call (GOSUB) in the program.
1003	Returns a dynamic array containing a list of open files. Each field has two values; the first holds the internal file number, the second holds the file's pathname.
1004	Returns the peak number of files that have been open at one time since QM was started.
1005	Returns the combined date and time value as DATE() * 86400 + TIME().
1006	Returns 1 if running on a Windows NT style system (NT and later).
1007	Returns the current transaction number, zero if not in a transaction.
1008	Returns the current transaction level, zero if not in a transaction.
1009	Returns the system byte ordering, 1 for high byte first, 0 for low byte first.
1010	Returns the platform name; Windows, Linux or FreeBSD.
1011	Returns the pathname of the QM configuration file.
1012	Returns QM version number.
1013	Returns user limit, excluding users reserved for phantom processes.
1014	Returns user limit, including users reserved for phantom processes.
1015	Returns the name of the host computer system.
1016	Returns the remaining number of licensed non-phantom users.
1017	Returns the tcp/ip port number for a socket connection.
1018	Returns the device licensing connection limit.
1019	Returns the offset in seconds of the user's local time zone from UTC. This will be negative for zones to the west of longitude zero.
1020	Returns the time of day in milliseconds since midnight UTC.
1022	Returns the unique qmid value for a user mode installation of QM, zero for a standard installation.
1023	Returns true on 64 bit systems, false on 32 bit systems.
1024	Returns the current working directory pathname when QM was entered.
1025	Returns a dynamic array where field 1 is a multivalued list of environment variable names and field 2 is a corresponding list of their values.
1026	Returns *xxx* when QM is entered using "qm *xxx*".
1027	Returns the name of the serial port when logged in on a serial connection.
1028	Returns the system id of the active QM licence, zero if the licence is not system specific.
1029	Returns the current internal subroutine depth.
1030	Returns login time as date * 86400 + time in the user's local time zone using Pick date numbering.
1031	Returns operating system process id.
1032	Returns and clears the break pending flag, set if the break key is pressed with breaks disabled.

1033	Returns true if a COMO file is active.
1034	Returns true if COMO output is active but suspended.
1035	Returns the time as seconds since 00:00:00 on January 1 1970, Universal Coordinated Time (UTC). Same as the EPOCH() function.
1036	Returns the period in tenths of a second for which login was delayed waiting for a spare user. See the LGNWAIT configuration parameter.
1037	Returns a list of files open to each user.
1038	Returns login time as an epoch value.
1040	Returns true if operating system filenames are case insensitive.
1041	Returns a field mark delimited list of active replication subscribers.
1042	Returns the system id of the server. This will differ from SYSTEM(1028) for a USB installation or a licence that is not system specific.
1043	Returns the current CSV mode that determines quoting rules.
1045	Returns time at which QM was started as an epoch value.
1046	Returns the current CSV delimiter character set by CSV.MODE.

TAN(*expr***)**
Returns the tangent of a value.

TCLREAD *var*
Returns the sentence that started the current program (Similar to use of as @SENTENCE).

TERMINFO()
TERMINFO(*cap.name***)**
Returns information from the terminfo database.

TESTLOCK(*lock.num***)**
Returns the owner of a task lock, zero if free.

TIME()
Returns the current time as the number of seconds since midnight.

TIMEDATE()
Returns the current time and date as a 20 character string in the form hh:mm:ss dd mmm yyyy

TIMEOUT *fvar***,** *interval*
Sets a timeout for **READBLK** and **READSEQ**.

TOTAL(*expr***)**
Accumulates totals for use with the **CALC** query processor keyword (Only in dictionary I-type items).

TRANS({**DICT**} *file.name***,** *record.id***,** *field***,** *action***)**
Returns a field or the entire record from a named data file. Normally only used in dictionary I-type items.

TRANSACTION ABORT
Aborts a transaction.

TRANSACTION COMMIT {**THEN** *statements*} {**ELSE** *statements*}
Commits a transaction.

TRANSACTION START {**THEN** *statements*} {**ELSE** *statements*}
Starts a transaction.

TRIM(*string***)**
Returns a copy of *string* with all leading and trailing spaces removed and multiple embedded spaces compressed to a single space.

TRIM(*string***,** *character* **{,** *mode***})**
Removes excess characters from a string. *Mode* evaluates to a single character which determines the mode of trimming:

A   Remove all occurrences of *character*.
B   Remove all leading and trailing occurrences of *character*.
C   Replace multiple instances of *character* with a single *character*.
D   Remove all leading and trailing spaces, replacing multiple embedded spaces with a single space. The value of *character* is ignored.
E   Remove all trailing spaces. The value of *character* is ignored.
F   Remove all leading spaces. The value of *character* is ignored.
L   Remove all leading occurrences of *character*.
R   Remove all leading and trailing occurrences of *character*, replacing multiple embedded instances of *character* with a single *character*.
T   Remove all trailing occurrences of *character*.

TRIMB(*string***)**
Removes trailing spaces from a string.

TRIMBS(*dynarray***)**
Returns a dynamic array with trailing spaces removed from each element of *dynarray*.

TRIMF(*string***)**
Removes leading spaces from a string.

TRIMFS(*dynarray***)**
Returns a dynamic array with leading spaces removed from each element of *dynarray*.

TRIMS(*dynarray***)**
Returns a dynamic array with all leading and trailing spaces removed and multiple embedded spaces compressed to a single space in each element of *dynarray*.

TRIMS(*string***,** *character* **{,** *mode***})**
Removes excess characters from a each element of a dynamic array. *Mode* evaluates to a single character which determines the mode of trimming:

A   Remove all occurrences of *character*.
B   Remove all leading and trailing occurrences of *character*.
C   Replace multiple instances of *character* with a single *character*.
D   Remove all leading and trailing spaces, replacing multiple embedded spaces with a single space. The value of *character* is ignored.
E   Remove all trailing spaces. The value of *character* is ignored.
F   Remove all leading spaces. The value of *character* is ignored.
L   Remove all leading occurrences of *character*.
R   Remove all leading and trailing occurrences of *character*, replacing multiple embedded instances of *character* with a single *character*.
T   Remove all trailing occurrences of *character*.

TTYGET()
Returns a dynamic array containing the current terminal settings.
Field   Content

1	Ctrl-C treated as the break key? (PTERM BREAK mode)
2	Case inversion on? (PTERM CASE mode)
3	Break character value (PTERM BREAK n)
4	Output newline sequence (PTERM NEWLINE)
5	Input return key code (PTERM RETURN)

TTYSET *var*
Sets the terminal modes.

UNASSIGNED(*var***)**
Tests whether a variable is unassigned.

UNLOCK {*lock.num*} {**THEN** statement(s)} {**ELSE** *statement(s)*}
Releases one or all of 64 system wide task locks.

UNTIL *expr*
Used in conjunction with the **FOR** / **NEXT** or **LOOP** / **REPEAT** constructs to determine whether execution of the loop should continue.

UPCASE(*string***)**
Returns a string with all letters converted to upper case.

VOCPATH(*filename* {**,** *dict.flag*}**)**
Uses the VOC to return the pathname of the supplied *filename*.

VOID *expr*
Evaluates an expression and discards the result.

VSLICE(*string***,** *vpos***)**
Returns a string formed by extracting a given value position from a dynamic array.

VSLICE(*string***,** *vpos***,** *svpos***)**
Returns a string formed by extracting a given subvalue position from a dynamic array.

WAIT.FILE.EVENT(*event***,** *timeout***)**
Wait for a file monitoring event.

WAKE *user.no*
Resume execution of another process that has executed a **PAUSE**.

WEOFSEQ *file.var* {**ON ERROR** *statement(s)*}
Truncates a record open for sequential access at the current position.

WHILE *expr*
Used in conjunction with the **FOR** / **NEXT** or **LOOP** / **REPEAT** constructs to determine whether execution of the loop should continue.

WRITE *var* **TO** *file.var***,** *record.id* {**ON ERROR** *statement(s)*}
Writes a record to a previously opened file.

WRITEU *var* **TO** *file.var***,** *record.id* {**ON ERROR** *statement(s)*}
Writes a record to a previously opened file, preserving any lock on the record.

WRITE.SOCKET(*skt*, *data*, *flags*, *timeout*)
Writes data to a socket.

WRITEBLK *var* **TO** *file.var*
    {**ON ERROR** *statement(s)*}
    {**THEN** *statement(s)*}
    {**ELSE** *statement(s)*}
Writes data at the current file position in a record previously opened using **OPENSEQ**.

WRITECSV *var1*, *var2*, ... **TO** *file.var*
    {**ON ERROR** *statement(s)*}
    {**THEN** *statement(s)*}
    {**ELSE** *statement(s)*}
Writes CSV format data at the current file position in a record previously opened using **OPENSEQ**.

WRITESEQ *var* **TO** *file.var* {**ON ERROR** *statement(s)*}
    {**THEN** *statement(s)*}
    {**ELSE** *statement(s)*}
Writes a string array to a directory file record previously opened for sequential access.

WRITESEQF *var* **TO** *file.var* {**ON ERROR** *statement(s)*}
    {**THEN** *statement(s)*}
    {**ELSE** *statement(s)*}
Writes a string array to a directory file record previously opened for sequential access, flushing the data to disk before continuing program execution.

WRITEV *var* **TO** *file.var*, *record.id*, *field.expr*
    {**ON ERROR** *statement(s)*}
Writes a specific field to a record of a previously opened file.

WRITEVU *var* **TO** *file.var*, *record.id*, *field.expr*
    {**ON ERROR** *statement(s)*}
Writes a specific field to a record of a previously opened file, preserving any record lock.

XLATE({**DICT**} *file.name*, *record.id*, *field*, *action*)
Returns a field or the entire record from a named data file. Normally only used in dictionary I-type items.

XTD(*expr*)
Converts the supplied hexadecimal string in *expr* to a number.

10   Standard QMBasic Subroutines

!ABSPATH(*path***,** *dir***,** *file***)**
Forms an absolute pathname from a directory and file path.

!ACCOUNT.RULES(*username***,** *accounts***,** *barred***)**
Returns a list of accounts that the specified user may or may not enter.

!ATVAR(*value***,** *name***)**
Retrieves the value of an @-variable.

!ERRTEXT(*errno***)**
Returns the text description for a QM error number.

!GETPU(*key***,** *unit***,** *value***,** *status***)**
Gets the characteristics of a print unit. The *key* value is as for the **GETPU()** function.

!LISTU(*dyn.array***)**
Returns raw data as used by the **LISTU** command.

!PARSER(*key***,** *type***,** *string***,** *keyword* {**,** *voc.rec*}**)**
Parses a command line.

!PATHTKN(*outpath***,** *inpath***)**
Processes special tokens in a VOC or ACCOUNTS file pathname.

!PICK(*item***,** *top.line***,** *item.list***,** *title***,** *pos***)**
Displays a list of entries from which a user may select one.

!PICKLIST(*value***,** *list***,** *return.col***,** *index.col***)**
Displays a list of entries from which a user may select one.

!SCREEN(*scrn***,** *data***,** *step***,** *status***)**
Performs screen based input using a screen definition created using the **SCRB** command.

!SETPU(*key***,** *unit***,** *value***,** *status***)**
Sets the characteristics of a print unit. The key value is as for the **SETPU()** function.

!SETVAR(*name***,** *value***)**
Sets the value of a user defined @-variable.

!SORT(*in.list***,** *out.list***,** *sort.rule***)**
Sorts the elements of a dynamic array according to a specified sorting rule.

!USERNAME(*name***,** *userno***)**
Returns the user login name for a given QM user number.

!USERNO(*userno***,** *name***)**
Returns the QM user numbers for users logged in with the given user name.

!VOCREC(*rec***,** *id***)**
Reads a VOC record, following remote record pointers.

PCL.xxx() Functions

These all map onto calls to the **!PCL()** subroutine via definitions in the PCL.H include record. See the *QM Reference Manual* or the online help for details of the arguments to these functions.

PCL.BOX(*left***,** *top***,** *width***,** *height***,** *pen.width***,** *radius***)**
> Draws a rectangular box.

PCL.COPIES(*copies***)**
> Sets the number of copies to be printed.

PCL.CURSOR(*x***,** *y***)**
> Sets the current cursor position.

PCL.DUPLEX(*mode***)**
> Sets duplex mode. 0 = off, 1 = long edge binding, 2 = short edge binding.

PCL.FONT(*font***)**
> Sets the font details for text output.

PCL.HLINE(*x***,** *y***,** *length***,** *pen.width***)**
> Draws a horizontal line.

PCL.ORIENTATION(*layout***)**
> Specifies the page format.

PCL.PAPER.SIZE(*size***)**
> Specifies the page size.

PCL.RESET()
> Resets the printer.

PCL.RESTORE.CSR()
> Restores a previously saved cursor position.

PCL.SAVE.CSR()
> Saves the current cursor position.

PCL.VLINE(*x***,** *y***,** *length***,** *pen.width***)**
> Draws a vertical line.

11   QMBasic Debugger

Function Key Commands

F1	Display help screen
F2	Abort program
F3	Stop program
F4	Display user screen (normal program output)
F5	Free run
F6	Step over subroutine call (internal or external)
F7	Step program element
F8	Step line
Ctrl-F7	Exit subroutine, returning to parent program, internal or external subroutine
Ctrl-F8	Exit program, returning to parent program or external subroutine

Word Based Commands (uppercase letters are the short form where appropriate)

ABORT	Quit the program, generating an abort.
BRK {*n*}	Set a breakpoint on line *n*. Shows breakpoints if *n* omitted.
CLR	Clear all breakpoints.
CLR *n*	Clear breakpoint on line *n*.
DUMP *var path*	Dumps a variable to an operating system level file.
EP	Exit program, returning to parent program or external subroutine.
EXit	Exit subroutine, returning to parent program, internal or external subroutine.
Goto *n*	Continue execution at line *n*.
HELP	Display help page.
PDUMP	Generate a process dump file.
Quit	Quit the program, generating an abort
Run	Free run.
Run *n*	Run to line *n*.
SET *var = value*	Change content of a program variable
STACK	Display call stack, most recent program first.
Step *n*	Execute *n* lines.
Step .*n*	Execute *n* elements.
StepOver	Step over subroutine call (internal or external)
STOP	Quit the program, generating a stop.
UnWatch	Cancels an active watch action.
View	Display user screen (normal program output)
Watch {*var*}	Watches the named variable. Shows watched variable if *var* omitted.
Watch *var op value*	Watches the named variable, stopping when the condition is met.
XEQ *command*	Execute *command*.

The following commands apply only to full screen mode debugging:

SRC	Revert to default program source display
SRC *name*	Show source of program *name*.
SRC *n*	Display around line *n* of currently displayed program.
SRC +*n*	Move display forward *n* lines in program.
SRC -*n*	Move display backward *n* lines in program.
WINdow *n*	Sets the number of lines in the command area of the screen.

The following commands apply only to non-full screen mode debugging:

SRC	Display current source line
SRC *n*	Display source line *n*. Entering a blank debugger command line after this command will display the next source line.
SRC *n,m*	Display *m* lines starting at source line *n*. The value of *m* is limited to three lines less than the screen size. Entering a blank debugger command line after this command will display the next *m* source lines.

Displaying Variables

*var***/**	Displays the content of variable *var*
var(*index*)**/**	Displays an element of dimensioned matrix *var*
var<*f,v,s*>**/**	Displays a field, value or subvalue of variable *var*
/*	Displays all variables (full screen mode only)

Alternative syntaxes of /*var*, *var*? and ?*var* are provided for compatibility with other environments.

Entering a blank command line repeats the last command.

12   QMClient API

Error Code Values

0	SV_OK	Action successful
1	SV_ON_ERROR	Action took the ON ERROR clause to recover from a situation that would otherwise cause the server process to abort.
2	SV_ELSE	Action took the ELSE clause. In most cases the **QMStatus()** function can be used to determine the error number.
3	SV_ERROR	An error occurred for which extended error text can be retrieved using the **QMError()** function.
4	SV_LOCKED	The action was blocked by a lock held by another user. The **QMStatus()** function can be used to determine the blocking user.
5	SV_PROMPT	A command executed on the server is waiting for input. The only valid client functions when this status is returned are **QMRespond()**, **QMEndCommand** and **QMDisconnect**.

QMCall *SubrName***,** *ArgCount* {**,** *ArgList* }
   Calls a catalogued subroutine on the server.

Text = **QMChange(***Src***,** *OldStr***,** *NewStr***,** *Occurrences***,** *Start***)**
   Replaces occurrences of one substring with another in a string.

Value = **QMChecksum(***String***)**
   Calculate checksum value for given string.

QMClearSelect *ListNo*
   Clears a select list.

QMClose *FileNo*
   Closes a file.

Ok = **QMConnect(***Host***,** *Port***,** *UserName***,** *Password***,** *Account***)**
   Establishes a QMClient session on a named system.

Ok = **QMConnectLocal(***Account***)**
   Establishes a QMClient session on the local system as the current user.

Ok = **QMConnected()**
   Confirms whether a QMClient session is open.

QMConnectionType *Type*
   Sets parameters that affect the behaviour of QMClient.

Ct = **QMDcount**(*Src*, *Delim*)
Counts delimited items in a string.

Text = **QMDecrypt**(*Data*, *KeyString*)
Decrypt data.

DynArray = **QMDel**(*Src*, *Fno*, *Vno*, *Svno*)
Deletes a field, value or subvalue from a dynamic array.

QMDelete *FileNo*, *Id*
Deletes a record from a file.

QMDeleteu *FileNo*, *Id*
Deletes a record from a file, retaining the record lock.

QMDisconnect
Terminates a QMClient session.

QMDisconnectAll
Terminates all open QMClient sessions.

Text = **QMEncrypt**(*Data*, *KeyString*)
Encrypt data.

QMEndCommand
Aborts a command executed on the server that is requesting input.

Text = **QMError**()
Returns extended error message text.

Text = **QMExecute**(*Cmnd*, *Errno*)
Executes a command on the server.

Text = **QMExtract**(*Src*, *Fno*, *Vno*, *Svno*)
Extracts a field, value or subvalue from a dynamic array.

Text = **QMField**(*Src*, **Delimiter**, *Start*, *Occurrences*)
Extracts one or more components of a delimited string.

QMFree(*addr*)
Releases dynamic memory returned by other API functions (C API only).

Session = **QMGetSession**()
Returns the session number for the currently selected QMClient session.

Text = **QMGetVar**(*Name*)
Returns the value of an @-variable from the server.

DynArray = **QMIns**(*Src*, *Fno*, *Vno*, *Svno*, *NewData*)
Inserts a field, value or subvalue in a dynamic array.

Found = **QMLocate**(*Item*, *DynArray*, *Fno*, *Vno*, *Svno*, *Pos*, *Order*)
Searches a dynamic array for a field, value or subvalue matching a given string.

Ok = **QMLogto**(*Account*)
   Moves to an alternative account.

QMMarkMapping *FileNo*, *State*
   Enables or disable mark mapping for a directory file.

Bool = **QMMatch**(Src, *Pattern*)
Matches a character string against a pattern template.

Text = **QMMatchfield**(*Src*, *Pattern*, *Component*)
   Matches a character string against a pattern template and extracts the part corresponding to a
   specified pattern component.

List = **QMNextPartial** *ListNo*
   Returns next part of a select list built using QMSelectPartial().

FileNo = **QMOpen**(*FileName*)
   Opens a file.

Text = **QMRead**(*FileNo*, *Id*, *Errno*)
   Reads a record without locking.

Text = **QMReadl**(*FileNo*, *Id*, *Wait*, *Errno*)
   Reads a record with a shareable read lock.

Text = **QMReadList**(*ListNo*, *Errno*)
   Reads a select list into a dynamic array in the client application

Text = **QMReadNext**(*ListNo*, *Errno*)
   Retrieves the next entry from a select list

Text = **QMReadu**(*FileNo*, *Id*, *Wait*, *Errno*)
   Reads a record with an exclusive update lock.

QMRecordlock *FileNo*, *Id*, *UpdateLock*, *Wait*
   Locks a record.

QMRelease *FileNo*, *Id*
   Releases a record lock.

DynArray = **QMReplace**(*Src*, *Fno*, *Vno*, *Svno*, *NewData*)
   Replaces the content of a field, value or subvalue in a dynamic array.

Text = **QMRespond**(*Response*, *Errno*)
   Responds to a request for input from a command executed on the server.

Text = **QMRevision**()
   Returns the revision numbers of the client and server components.

QMSelect *FileNo*, *ListNo*
   Generates a select list containing the ids of all records in a file.

QMSelectIndex *FileNo*, *IndexName*, *IndexValue*, *ListNo*
　　Generates a select list from an alternate key index.

Key = **QMSelectLeft** *FileNo*, *IndexName*, *ListNo*
　　Scan left in an alternate key index.

List = **QMSelectPartial** *FileNo*, *ListNo*
　　Builds a select list and returns multiple record ids.

Key = **QMSelectRight** *FileNo*, *IndexName*, *ListNo*
　　Scan right in an alternate key index.

Key = **QMSetLeft** *FileNo*, *IndexName*
　　Position at the extreme left in an alternate key index scan.

Key = **QMSetRight** *FileNo*, *IndexName*
　　Position at the extreme right in an alternate key index scan.

Bool = **QMSetSession**(*Session*)
　　Selects an open QMClient session.

Code = **QMStatus**()
　　Returns the value of the QMBasic **STATUS**() function for the last server function executed.

QMTrapCallAbort *Mode*
　　Enables or disables client side trapping of aborts in QMCall.

QMWrite *FileNo*, *Id*, *Rec*
　　Writes a record.

QMWriteu *FileNo*, *Id*, *Rec*
　　Writes a record, retaining the lock.

13 Conversion Codes

A - A-correlative
A;*expr*   Executes A-correlative expression *expr* interpretively.

B - Boolean
Input: Accepts N and Y, upper or lowercase, returning 0 for N and 1 for Y.
Output: Returns N for false, Y for true.

B64 - Base64
Input: Converts from base64 encoding.
Output: Converts to base64 encoding.

C - Concatenation
C{;} *c expr c expr …*

c is separator, semicolon if no separator is required.
expr is data item (field number, quoted string or asterisk for data being converted.

D - Date
D {*y*} {*c*} {*fmt*} {[*f1, f2, f3, f4, f5*]}

y	Number of digits to appear in the year number (Default 4).
c	Character to separate year, month and day components of the converted date. Defaults to a space. Set *c* as the digit zero for no separator.
fmt	Specifies components to be present in the converted date:

	D	Day of month.
	DO	Ordinal day of month (1st, 2nd, 3rd, etc)
	E	Toggles European date format.
	J	Julian date (days since the start of the year).
	L	Alphabetic month and day names to appear with only first character in uppercase.
	M	Month in format determined by format modifiers.
	MA	Month name.
	Q	Quarter number (1 to 4).
	W	Day of week number. Monday is day 1.
	WA	Weekday name.
	WI	ISO week number.
	X	ANSI X12 format date (YYYYMMDD with no separators).
	Y	Year.
	YI	ISO year number.
[*f1,f2,f3,f4,f5*]	Format modifiers:	
	n	Use *n* characters.
	A	Display as alphabetic (applies to month component only).
	A*n*	Display as *n* alphabetic characters (applies to month component only).
	Z	Suppress leading zeros.
	Z*n*	Display as *n* digits with leading zeros replaced by spaces.
	"*text***"**	Uses *text* as separator after associated item.

Special codes commencing "ISO8601" perform ISO 8601 format conversions.

E - Epoch
E {*y*} {*c*} {*fmt*} {[*f1*, *f2*, *f3*, *f4*, *f5*]}

y	Number of digits to appear in the year number (Default 4).
c	Character to separate year, month and day components of the converted date. Defaults to a space. Set *c* as the digit zero for no separator.
fmt	Specifies components to be present in the converted date:

A	ASCII format date/time (e.g. Thu 16 Apr 15:02:21 2009). No other elements may be included in the same conversion.
D	Day of month.
DO	Ordinal day of month (1st, 2nd, 3rd, etc)
E	Toggles European date format.
J	Julian date (days since the start of the year).
L	Alphabetic month and day names to appear with only first character in uppercase.
M	Month in format determined by format modifiers.
MA	Month name.
O{:}	Offset from GMT in form +*hh*:*mm* or -*hh*:*mm*. Colon is shown if included in code.
Q	Quarter number (1 to 4).
T{**H**}{**S**}{*s*}	Time. Elements are as for MT conversion.
W	Day of week number. Monday is day 1.
WA	Weekday name.
WI	ISO week number.
X	ANSI X12 format date (YYYYMMDD with no separators).
Y	Year.
YI	ISO year number.
Z	Time zone code

[*f1*,*f2*,*f3*,*f4*,*f5*,*f6*,*f7*]	Format modifiers:
n	Use *n* characters.
A	Display as alphabetic (applies to month component only).
A*n*	Display as *n* alphabetic characters (applies to month component only).
Z	Suppress leading zeros.
Z*n*	Display as *n* digits with leading zeros replaced by spaces.
"*text***"**	Uses *text* as separator after associated item.

Special codes commencing "ISO8601" perform ISO 8601 format conversions.

F - F-correlative
F;*expr*	Executes F-correlative expression *expr* interpretively.

G - Group
G*cn*	Returns the first *n* components of the source data delimited by character *c*.
G*scn*	Skips s components and then returns the next n components of the source data delimited by character *c*.

IS, IL - Integer
IS	Converts a QMBasic integer value to a hardware specific short integer (16 bits).

IL	Converts a QMBasic integer value to a hardware specific long integer (32 bits).
ISL, ILL	As **IS** and **IL** but forcing use of a low byte first representation.
ISH, ILH	As **IS** and **IL** but forcing use of a high byte first representation.

L - Length

L	Returns the length of the string being converted.
L*n*	Returns the original string if its length is less than or equal to *n*. Otherwise it returns a null string.
L*n*,*m*	Returns the original string if its length is greater than or equal to *n* and less than or equal to *m*. Otherwise it returns a null string.

MC - Character

MCA	Delete all non-alphabetic characters
MC/A	Delete all alphabetic characters
MCAN	Delete all non-alphanumeric characters
MC/AN	Delete all alphanumeric characters
MCL	Convert to lower case
MCN	Delete all non-numeric characters
MC/N	Delete all numeric characters
MCP	Replace non-printing characters by dots
MCS	Convert the first alphabetic character of each sentence to uppercase
MCT	Capital initial all words
MCU	Convert to uppercase

MD, ML, MR - Masked Decimal

MD *n* {*f*} {,} {*$*} {*s*} {[*intl*]} {**P**} {**Z**} {**T**}
ML *n* {*f*} {,} {*$*} {*s*} {[*intl*]} {**P**} {**Z**} {**T**} {*x*{*c*}} {*fx*}
MR *n* {*f*} {,} {*$*} {*s*} {[*intl*]} {**P**} {**Z**} {**T**} {*x*{*c*}} {*fx*}

n	The number of digits to appear to the right of the decimal point.
f	The position of the implied decimal point in the data to be converted. If omitted, *f* defaults to the same value as *n*.
,	Inserts commas between every third digit to the left of the decimal point.
$	Specifies a currency prefix.
s	Specifies the handling of the numeric sign of the value.

	+	places a + or - sign to the right of the converted data.
	-	places a - sign to the right of negative values or a space to the right of positive values.
	(	encloses negative values in round brackets. A positive value has a space placed to its right.
	<	encloses negative values in angle brackets. A positive value has a space placed to its right.
	C	places the letters cr to the right of negative values or two spaces to the right of positive values. Use the **CRDB.UPCASE** keyword of the **OPTION** command to change this to CR.
	D	places the letters db to the right of negative values or two spaces to the right of positive values. Use the **CRDB.UPCASE** keyword of the **OPTION** command to change this to DB.

[*intl*]	Up to four comma separated items which specify the prefix, thousands separator, decimal separator and suffix to be applied to the converted value.

Z   specifies that a zero value should be represented by a null string on output conversion.

T   specifies that trailing decimal places should be truncated rather than rounded on output conversion.

x{c}   specifies the field width and fill character.

fx   specifies padding. *f* is * for asterisk, # for space, % for zero. *x* is the field width. This element can be a complete format mask (e.g. #3-#4). When the PICK.ML.CONV.MASK mode of the OPTION command is in effect, the *fx* element may be enclosed in round brackets in ML and MR conversions.

The *x{c}* and *fx* padding specifications cannot be used together.

MT - Time

MT {**M**} {**H**} {**S**} {*c*}

M specifies that the time value is in milliseconds (output conversion only).

H specifies that 12-hour format with either AM or PM appended is to be used.

S specifies that output conversion is to include the seconds (and possibly milliseconds) element.

c is the character to separate the hours, minutes and seconds element.

MB, MO, MX - Radix

MB{**0C**} Binary conversion

MO{**0C**} Octal conversion

MX{**0C**} Hexadecimal conversion

MCD{X}, MCX{D} - Radix

MCD{**X**}   Decimal to hexadecimal (output), hexadecimal to decimal (input)

MCX{**D**}   Hexadecimal to decimal (output), decimal to hexadecimal (input)

P - Pattern Matching

P(*template*){**;**(*template*)...}

Attempts to match the input data against each *template* in turn. If a match is found, the original data is returned. Otherwise a null string is returned.

R - Range Check

R*n*,*m*{**;***n,m*...}

Tests whether the input data is an integer value in the range *n* to *m*. If so, the input data is returned. If not, a null string is returned.

S - Substitution

S;*value1***;***value2*

Returns a value determined by *value1* if the source data is not zero or null, a value determined by *value2* if the source data is zero or null. The *value1* and *value2* items may be:

0   Returns the value of @ID

number   Returns the specified field from @RECORD

*   Returns the source data unchanged

'*text*'   Returns the given *text* string. The string may be enclosed in single quotes, double quotes or backslashes.

T*file* - File Translation

T*file*;*cv*;*i*;*o*

Uses the source data as the id of a record in *file* and returns data from this record.

file is the file name. Prefix with DICT or * to reference the dictionary.

c	identifies the action if the record does not exist or the field is null.
C	returns the record id.
V	displays an error message and returns a null value.
X	returns a null value.
I	Like V for input conversion, C for output conversion
O	Like C for input conversion, V for output conversion
v	value position of data to return. If omitted, all values are returned with value and subvalue marks replaced by spaces.
i	field position of data to return on input conversion
o	field position of data to return on output conversion

T - Text substring

T*m*{**,***n*}

Returns *n* characters starting at character *m* of the source data. If *n* is omitted, the last *m* characters of the source data are returned.

U - User Defined Conversions

U*subrname*

U*subrname***.***extension* *extension* is available in @CONV variable.

These codes call a catalogued subroutine named *subrname*. The format of this is:

CONV.SUBR(*result*, *src*, *status*, *oconv*)

result should be set by the subroutine as the result of the conversion.

src   is the item to be converted.

status is the value to be set for the **STATUS()** function.

oconv indicates whether this is an input (0) or output (1) conversion.

<*f*,*v*,*s*> - Field Extraction

<f >	Extract field *f*.
<f,v >	Extract field *f*, value *v*.
<f,v,s>	Extract field *f*, value *v*, subvalue *s*.

14   Format codes

{*field.width*} {*fill.char*} *justification* {*n*{{*m*} {*conv*} {*mask*}

field.width	Width of the formatted data. If omitted, *mask* must be specified.
fill.char	Character to be appear in unused positions of the result string.
justification	alignment mode to be applied: **C** Centered. **L** Left justification. **R** Right justification. **T** Text justification. **U** Left justification. In the query processor, data wider than *field.width* extends into the space to its right, possibly overwriting whitespace in later columns.
n	The number of decimal places to appear in the result when formatting numeric data.
m	The scaling factor to be applied. The value being formatted is scaled by moving the decimal point *m* - *p* places to the left where *p* is the current precision value.
conv	Any meaningful combination of the following codes: **$** Add a currency symbol prefix. **,** Insert commas as thousands delimiters when converting numeric data. **B** Append db to negative numbers, two spaces to positive numbers. Use the **CRDB.UPCASE** keyword of the **OPTION** command to change this to DB. **C** Append cr to negative numbers, two spaces to positive numbers. Use the **CRDB.UPCASE** keyword of the **OPTION** command to change this to CR. **D** Append db to positive numbers, two spaces to negative numbers. Use the **CRDB.UPCASE** keyword of the **OPTION** command to change this to DB. **E** Enclose negative number in angle brackets (<…>). Positive numbers are followed by a single space. **M** Append a minus sign to negative numbers. **N** Suppress any sign indicator. **Z** Zero should be represented by a null string.
mask	A mask to be used to format the data. If omitted, *field.width* must be specified. Both can be used together. The mask consists of a character string containing #, * or % characters and other characters. Each #, * or % is substituted by one character from the source data. Other characters are copied directly to the result string. Multiple #, * or % characters may be represented by a single #, * or % followed by a number indicating the number of characters to be inserted. Characters having special meaning within the format string may be prefixed by a backslash (\) to indicate that they are to be treated as text.

Where the *mask* specifies more characters than in the data being converted, positions corresponding to # characters in the mask are replaced by the *fill.char* , positions corresponding to * characters in the mask are replaced by asterisks and positions corresponding to % characters in the mask are replaced by zeros. If the data is left aligned, the padding is inserted in the rightmost positions. If the data is right aligned, the padding is inserted in the leftmost positions.

If the *mask* specifies fewer characters than in the data being converted, part of the source data will be lost. A left aligned format will truncate the source data and a right aligned format will lose data from the start of the source.

15   ED Commands

ED {**DICT**} *file.name* {*record.id*} {**NO.QUERY**}

Positioning Commands

n	Move to line *n*.
+*n*	Move forwards *n* lines.
-*n*	Move backwards *n* lines.
B	Move to the last line of the record.
F{*n*} {*string*}	Move forward to the next line containing *string* starting at column position *n* (from one).
G*n*	Move to line *n*.
G<	Move to the first line of the currently defined block.
G>	Move to the last line of the currently defined block.
L {*string*}	Move forward to the next line containing *string*.
M {*pattern*}	Move forward to the next line matching *pattern*.
PO*n*	Move to line *n*.
T	Move to before line 1.

Displaying Text

L*n*	Display *n* lines (Same as **P** except that *n* must be included).
P{*n*}	Display *n* lines, moving the current line forward to the final displayed line.
PL{{**-**}*n*}	Display *n* lines relative to the current line position without changing the current line position.
PP{*n*}	Display *n* lines surrounding the current line position.

Inserting Text

I	Enter insert mode under the current line.
I *text*	Insert *text* as a single line under the current line.
IB	Enter insert mode above the current line.
IB *text*	Insert *text* as a single line above the current line.
LOAD {{*filename*} *record.id*}	Insert part or all of the specified record

Deleting Lines

D{*n*}	Delete *n* lines starting at the current line position.
DE{*n*}	Delete *n* lines starting at the current line position.

Changing Text

A {*string*}	Append *string* to the current line.
B *string*	Break the current line into two after *string*.
C/*old.string*/*new.string*/{*n*}{**G**}{**B**}	Change *old.string* to *new.string* in the current line. The optional *n* component specifies that *n* lines starting at the current line are to be changed. **G** replaces all occurrences of *old.string*. **B** applies the change to the currently defined block.
CAT {*string*}	Concatenate the current line, *string* and the following line.
DUP {*n*}	Duplicate the current line *n* times.
R/*old.string*/*new.string*/{*n*}{**G**}{**B**}	Same as **C**.
R *text*	Replace the current line with the specified *text*.

Working with Multivalued Data

EV   Enters EV (edit values) mode for the current line. With dictionary I-types, **EV** breaks compound expressions to simplify editing.

QV   Exits EV mode, discarding any changes.

SV   Exits EV mode, saving any changes.

Block Edit Commands

<   Set the current line as the start line of the block.

>   Set the current line to be the end line of the block.

<>   Sets the block to be just the current line.

BLOCK Toggle block verification mode.

COPY   Copy the currently defined block to immediately after the current line position.

DROP   Delete the currently defined block.

MOVE   Move the currently defined block to immediately after the current line position.

PB   Display the currently defined block.

File Handling Commands and Leaving the Editor

DELETE   Delete the entire record from the file.

FD   Delete the entire record from the file.

FILE {{{**DICT**} *filename*} *record.id*}   Write the record being edited back to its file and exit.

FIB {{*filename*} *record.id*}   Write the record and compile it as a program and exit.

FIBR {{*filename*} *record.id*}   Write the record, compile it, run it as a program and exit.

N   Move to the next record in a select list.

QUIT   Exit, discarding changes.

EX   Exit, discarding changes.

SAVE {{{**DICT**} *filename*} *record.id*}   Write the record, remaining in the editor.

UNLOAD {{{**DICT**} *filename*} *record.id*}   Save part or all of the data as named record.

X   Abort a select list controlled edit, discarding changes to the current record.

Miscellaneous Commands

?   Display status information about the editor and the record being edited.

^   Toggle non-printing character expansion mode.

CASE OFF   Set case insensitive mode for the C, F and L commands.

CASE ON   Set case sensitive mode for the C, F and L commands.

COL   Display a column number ruler to aid alignment of inserted text.

FORMAT   Apply conventional indentation rules to the program being edited.

HELP {*topic*} Display a short description associated with the command identified by *topic*.

NPRINT   Toggles non-print character entry mode.

OOPS   Undo the most recent function that modified the record.

STAMP   Insert an edit history comment line below the current line.

SPOOL {*lines*} Print a copy of the record on the default printer.

XEQ {*command*}   Execute *command*.

Editor Command Stack

.A{*n*} *string*   Append *string* to entry *n* of the editor command stack.

.C{*n*}/*old.string*/*new.string*/   Change *old.string* to *new.string* in line *n* of the editor command stack.

.D{*n*}   Delete line *n* of the editor command stack.

.I{*n*} *string*   Insert *string* as entry *n* of the editor command stack.

.L{*n*}   List *n* lines of the editor command stack.

.R{*n*}   Recall line *n* of the editor command stack to the top of the stack.

.X{*n*}   Execute line *n* of the editor command stack.

Pre-stored Edit Commands

.S *item*	Save the most recent command
.S*n item*	Save command *n*
.S *item n***,***m*	Save commands *n* to *m*
.S *item n m*	Save commands *n* to *m*
.X *item*	Execute previously saved command.
.D *item*	Delete the specified item.
.L *item*	List the item.
.R *item*	Recall a previously saved set of commands to the stack.

LOOP *lineno count*   Jump back to *lineno*, *count* times.

16   SED Default Key Bindings

Editor commands in the default key bindings consist of keystrokes which are

- Control shift + key
- ESCape followed by another key
- Ctrl-X followed by another key
- Ctrl-X followed by control shift + key

	Ctrl-	**Esc-**	**Ctrl-X -**	**Ctrl-X Ctrl-**
A	Start line			
B	Back char	Back word	Goto buffer	*List buffers
C	Repeat	Capital init	*Quit	*Quit
D	Delete char	Delete word	*List records	Dive
E	End line	Run extension	*Execute macro	
F	Forward char	Forward word		*Find record
G	Cancel	Goto line		
H	Backspace			
I	Tab	Align text	Import	
J	Newline			
K	Kill line		Delete buffer	
L	Refresh	Lowercase		Lowercase region
M	Newline			
N	Down line	Next buffer		Nudge down
O	Overlay		Toggle window	
P	Up line	Previous buffer		Nudge up
Q	Quote char	Quote char	Query replace	
R	Reverse search	Reverse search	Replace	
S	Forward search	Forward search	Save record	Save record
T	Toggle chars			
U	Repeat	Uppercase	Up to parent	Uppercase region
V	Forward screen	Back screen	*View	
W	Delete region	Copy region	*Write record	*Write record
X	Ctrl-X prefix	*Command	*Export	Swap mark
Y	Insert kill buffer	Insert kill buffer		
Z	Up line			Nudge up
1			Unsplit window	
2			Split window	
(			*Start macro	
)			*End macro	
=			*Expand char	
.		Set mark		
<		Top		
>		Bottom		
Bkspc	Backspace	Back del word		
Del	Delete char			
Space		Close spaces		

Functions marked with an asterisk cannot be included in a macro and cannot be repeated using the **repeat** or **repeat count** functions.

17 Configuration Parameters

For full details, see the *QM Reference Guide*. Parameters marked with an asterisk can be modified for an individual process using the **CONFIG** command.

QM	CLEANUP	Interval in seconds between checks for abnormally terminated QM processes. The value must be in the range 30 to 3600 and defaults to 300 (five minutes) if this parameter is not present.
	CLIPORT	The port for incoming QMClient telnet connections.
*	CMDSTACK	Determines the default size of the command stack. The value must be in the range 20 to 999 and defaults to 99 if this parameter is not present.
*	CODEPAGE	Windows only. Specifies the code page to be used for QMConsole connections. If omitted, QM uses the default console code page. Note that a restriction in Windows requires that the console session is set to use Lucida Console font for this feature to work.
	DEADLOCK	Enable automatic deadlock detection.
*	DHCACHE	Sets the size of the dynamic file cache.
	ERRLOG	Sets the maximum size (kb) of the error log, zero to disable.
*	EXCLREM	Omits remote files from an ACCOUNT.SAVE.

*	FILERULE	Enables special filename syntaxes (additive values):
		  1 Allow *account*:*file*
		  2 Allow *server*:*account*:*file*
		  4 Allow PATH:*pathname*
		The CONFIG command can reduce this setting but not increase it.

	FIXUSERS	Defines a fixed range of user numbers that are reserved for users logging in with a specific user number (qm -*n*). FIXUSERS= *u*,*n* where *u* = lowest user number, *n* = number of user numbers to be reserved.
*	FLTDIFF	Controls floating point comparison.

*	FSYNC	Additive values determining when an fsync operation is performed to flush all updated data to disk:
		  1 Every time that a file's header is updated.
		  2 At transaction commit.
		  4 After every write.

*	GDI	Use GDI mode in SETPTR by default? (Windows)
*	GRPSIZE	Sets default group size used when creating a dynamic file.
*	INTPREC	Controls rounding when converting floating point values to integers.
	JNLDIR	Specifies the pathname of the directory used to store journalling log files. This directory will be created automatically if it does not exist.
	JNLMODE	Additive value specifying the journalling modes to be supported: 1 Enable before imaging (allows roll back)

		2 Enable after imaging (allows roll forward)
	JNLSIZE	Sets the approximate maximum size of a journalling log file in Mb. The default value is 10. The valid range is 1 to 512.
	LICENCE	Licence parameters. Not editable.
	LGNWAIT	Sets the maximum time in tenth of a second units that a process will wait if the licensed user limit has been reached.
*	LPTRHIGH	Sets default number of lines per page when a print unit is first referenced.
*	LPTRWIDE	Sets the default number of characters per line when a print unit is first referenced.
*	MAXCALL	Sets the maximum depth of nested CALLs including internal components of QM such as the command processor. If this limit is reached due to, for example, a program error resulting in a recursive call loop, QM will abort the program gracefully rather than failing in unpredictable ways when it runs out of memory. The value must be in the range 10 to 1000000 and defaults to 1000.
	MAXIDLEN	Sets the maximum length of a record id (63 to 255). Increasing this from its default value of 63 can cause compatibility problems.
*	MUSTLOCK	Enforce strict locking rules?
	NETFILES	Enable QMNet file access (additive values): 1 Allow access to remote files without use of QMNet. 2 Allow incoming QMNet connections.
	NUMFILES	Sets system wide limit on number of files that may be opened at one time.
	NUMLOCKS	Sets maximum number of record locks that can be held at one time.
*	OBJECTS	Limits number of programs that can be loaded into memory before discard is attempted.
*	OBJMEM	Limits size of all loaded object programs (kb) before discard is attempted.
	OPTIONS	Configurable features (additive values, QMSvc only): 1 Log client disconnection 2 Persistent shared memory 16 Do not log successful incoming telnet connection 32 Do not log successful incoming QMClient connection 64 Do not log successful incoming serial connection
	PDUMP	Controls security for The PDUMP command: 0 No restrictions. 1 Only administrators can dump processes running under other user names.
	PHANTOMS	Defines a fixed range of user numbers that are reserved for phantom processes started with the USER option to the PHANTOM command. PHANTOMS= u,n where u = lowest user number, n = number of user numbers to be reserved.
	PORT	The port for incoming client telnet connections.
	PORTMAP	Defines a fixed mapping between tcp/ip port numbers and QM

		user numbers. PORTMAP=p,u,n where p = lowest port number, u = lowest user number, n = number of ports/users to be mapped.
	PWDELAY	The pause in seconds between successive attempts to enter the user name and password on network connections. This parameter defaults to 5.
	QMCLIENT	Controls security for QMClient sessions: 0 No restrictions. 1 Bans use of QMOpen() and QMExecute(). 2 Limits users to calling subroutines compiled with the $QMCALL compiler directive. The CONFIG command can set new restrictions but not remove them.
	QMSYS	Sets location of QMSYS account directory.
*	RECCACHE	Sets the size of the record cache.
	REPLDIR	Pathname of directory to be used for replication log files on the publisher system. This directory will be created automatically when QM is started if it does not already exist.
	REPLMAX	Sets the limit on the number of simultaneous replication targets that may be active from a publisher. This parameter must be in the range 1 to 255 and defaults to 8.
	REPLPORT	Port number on which QM will listen for incoming network connections from a replication subscriber system. Use of port 4244 is recommended as this is the default port used by the subscriber.
	REPLSIZE	Sets the approximate maximum size of a replication log file in Mb. The default value is 10. The valid range is 1 to 512.
	REPLSRVR	Server name of a replication subscriber system. This must match the name used on the publisher when setting up replication.
	RETRIES	The maximum number of attempts allowed for entry of a valid username and password on network connections. This parameter defaults to 3.
*	RINGWAIT	Controls action when terminal input ring buffer is full (Windows).
*	SAFEDIR	Controls how directory file records are written.
	SECURITY	Sets system wide security options.
	SERIAL	Defines a serial port to be monitored for incoming QM connections. Multiple SERIAL parameters may be specified. The format is SERIAL=$port$,$rate$,$parity$,$bits$,$stop$ where $port$ = the port name (e.g. COM1) $rate$ = baud rate (e.g. 9600) $parity$ = none(0), odd(1), even(2) $bits$ = bits per byte (7 or 8) $stop$ = number of stop bits (1 or 2)
*	SH	(Not Windows) Determines the shell processor and its options to be used when the SH command is used to start an

		interactive shell. If not set, this parameter defaults to "/bin/bash -i" on Linux and "/bin/sh -i" on FreeBSD.
*	SH1	(Not Windows) Determines the shell processor and its options to be used when the SH command or the QMBasic OS.EXECUTE statement is used to execute a single command. If not set, this parameter defaults to "/bin/bash -c" on Linux and "/bin/sh -c" on FreeBSD.
*	SORTMEM	Sets size (kb) at which a sort switches from memory based to disk based.
*	SORTMRG	Sets number of streams for sort merge phase.
*	SORTWORK	Sets pathname of the directory to hold temporary sort workfiles.
*	SPOOLER	Sets the name of the default spooler on Linux and FreeBSD. If this parameter is not specified or is a null string, the lp spooler is used. The name specific may be another standard spooler (e.g. lpr) or a user written program or shell script to perform custom print management. The qualifying data to this configuration parameter can include other options to be passed to the selected spooler.
	SRVRLOG	Sets the maximum size of the log file (QMSvc only). A value of zero causes QMSvc to start a new log each time the service is started, saving the previous log as QMSvcLog.old. A non-zero value sets the maximum size of the QMSvc.log file in kb. When this size is reached, the first half of the data in the file is removed.
	STARTUP	Sets a command to be executed when QM starts. This may not contain double quotes. This parameter is not applicable to the PDA version of QM.
*	TEMPDIR	Sets pathname of the directory to hold temporary files.
*	TERMINFO	The pathname of the directory holding the terminfo database.
	TIMEOUT	The maximum wait period in seconds allowed during entry of a valid username and password on QMSvc connections. This parameter defaults to 30.
*	TIMEZONE	The timezone to be used by the epoch conversion code.
*	TXCHAR	Determines whether QM uses OEM character translation on terminal i/o (Windows only).
	UMASK	Sets the default umask value for direct telnet, QMClient and QMNet connections. The mask is specified as an octal value.
	USERPOOL	Determines the maximum user number that will be allocated by QM.
*	YEARBASE	Start of the 100 year range of dates entered with two digit year numbers.
QMTERM	ADDR	IP address (or host name) and port number of connection.