# The QEF Specialist's Reference Card (October, 2005)

## Qvrs Keywords: *qvrs-kws(x-qvrs)

**Begin** keyword to begin back-end or generator input
**addpath** keyword to add directories to a path
**append** keyword to append value to variable's current value
**bind** keyword to bind a variable's value to another variable's
**case** keyword specifying switch statement section
**cnilset** keyword to set variable but only if empty
**cset** keyword to set variable if not already set
**elif** the conditional else keyword
**else** the else keyword
**endpreamble** keyword to terminate a qeffile's preamble section
**endswitch** keyword to end a switch statement
**env_exports** keyword to name variables exported as environment variables
**fatal** keyword to output diagnostic message and abort
**fi** keyword that ends an if statement
**if** the if keyword
**include** keyword to include an argument file
**message** keyword to output non-fatal diagnostic message
**nextcase** keyword to advance to another case within the switch
**options** keyword to initialize list of option variables
**preamble** keyword to specify qeffile lines to be processed first
**prepend** keyword to insert argument at beginning of variable's value
**prereqvrs** keyword to load and set the Prereq variables
**psysvrs** keyword to load the @PsysVrs or p_@System[SysVrs].vrs file
**qef_imports** keyword to name variables imported by qefpp as symbols
**set** keyword to assign argument to a variable
**setifnil** keyword to set variable but only if empty
**suspend** keyword to suspend processing temporarily
**switch** keyword at beginning of a switch statement
**sysvrs** keyword to load the @SysVrs or @System[SysVrs].vrs file
**unset** keyword to undefine a variable
**unsetifnil** keyword to undefine a variable if it is empty
**varrepl** keyword to replace matched parts of a variable

## Qvrs Variables: *vars(x-qvrs)

**ARTMPDIR** name of ar(1)'s temporary directory
**AsSuffix** suffix of assembler files
**BeginLine** Begin line arguments
**Branch** path from RootDir to current qvrs file
**BuildHost** name of the host used for remote builds
**BuildPath** executable path used in builds
**BuildSys** type of the system on which build should be done
**CcAsFlag** cc flag to produce assembler file
**CcInclDirs** additional directories searched by cc for #include files
**CcLibDirs** list of directories searched for libraries by default
**ConfVrs** name of the configuration file
**ConfVrsFile** root name of the conf.vrs file
**ConfVrsPath** search path for @ConfVrs and @PsysVrs files
**ConfigName** name of the configuration
**CurDir** current working directory
**DEBUGGING** debugging option
**DOS** DOS option - set if building DOS product
**DefaultBeginLine** Default BeginLine setting
**Dots** path from current directory to current QvrsDir
**EditBranches** version control option forcing branches
**ExcludeDirs** list of directories ignored by treedirs(1)
**ExtraLibs** list of extra libraries to be linked with c modules
**FSIC_Except** bfsic exception file basename
**FSIC_FailsIfErrors** fsic fails on error option
**FSIC_Ftypes** file types used in FSIC files.fl file
**FSIC_Key** bfsic configuration key letter
**ForbidQef** set to forbid execution of qef in directory
**ForbidQefType** numeric type indicating how ForbidQef was set
**ForceExecution[backend]** back-end flag to force execution
**FullHostname** the full name of the host
**GCC** option specifying gcc being used
**GOT_NLS** option specifying nls system being used
**HtmlCommenting** use html commenting in qefpp
**INTERIX** build for and on the Interix system
**InCurDir** internal boolean specifying that QvrsDir == CurDir
**InclDirs[X11]** name of the X11 include directory
**InclPath** list of directories to be used as −I flags to cc
**InclsDbm** name of the master incls database
**InclsDbmTimeDirs** list of directories for which incls uses tines from database
**InclsDbmUpdate** name of the master incls database
**InstLog** pathname for instal(1) audit trail
**LIBS[file]** Libs line for file
**LastRoot** last directory of RootPath
**LibDirs[X11]** name of the X11 library directory
**LibMap[−lX]** mapping of library to other libraries
**LibMatch** prototype of file name used in library searches
**LibName** name of the library built within directory
**LibPath** list of directories searched for libraries
**LibRevs** revision number for shared libraries
**LibStatic[−lX]** suppress use of shared libraries for designated programs
**LibSuffix** suffix of object libraries
**Library[−lX]** explicit path name for a library
**Logname** name of logged in user
**MAN3DB_UPM** option specifying that man3 sections to be produced
**MAN3_EML** option specifying that eml3 sections to be produced
**MAN3_FORMED** option specifying that man3db sections piped through form
**MAXMKPROCS** maximum number of parallel back-end (e.g., mimk) processes
**MKCATMAN** option that catman directories are to be created
**MKSOLIBS** option that shared libraries can be constructed
**MkvernumFormat[X]** default format for mkvernum
**MkvernumPath** directories to be searched for mkvernum formats
**Modules[X]** specify modules linked with file X
**NO_LINT** option specifying that lint to be suppressed
**NO_MAN** option specifying that manuals not to be installed
**NoExeSuffix** suppress gratuitous .exe suffix
**NoRootVrs** boolean set to true if no root.vrs file found
**NoSfiles** no version admin files (e.g., sccs s-files) on this system
**NotSet** a magic read-only value used to unset variables
**ObjSuffix** suffix of an object file
**ObscuredLibs** list of obscured libraries
**OldRoot** the root path of obsolete or old source
**PROFILING** option specifying that profiling binaries to be produced
**PchExcludes** files that shouldn't be compiled with precompiled headers
**PermitQef** option to allow qefs in source trees
**PrereqConf[X]** prerequisite project's configuration
**PrereqFormat[X]** prerequisite project's sub-directory path formatting
**PrereqList** list of the project's prerequisite projects
**PrereqMap[X]** prerequisite project's version identifier
**PrereqPath[X]** directory path to be searched for prerequisite roots
**PrereqRoot[X]** root for a prerequisite project
**PrereqVersion[X]** prerequisite project's version
**PrereqVrs** name of the default prerequisite file to be read
**Project** name of the product
**PsysVrs** basename of qvrs file to be used as the psysvrs file
**PsysVrsFile** rooted name of the psysvrs file
**QTREE** built-in variable set to the location of the <qtree-root>
**QefAdm** name of the qef administration directory
**QefArgs** list of command line arguments to qef
**QefFile** name of the qeffile
**QefFile2** qeffile2 flag specified
**QefFile2[@RealBranch]** name of the qeffile2 for named directory
**QefFile[@RealBranch]** name of the qeffile for named directory
**QefPath** list of directories to be searched for qef header files
**QefSpecialProc** name of special qef −P process
**QefdirsHdr** qefdirs header file
**QremoteEnv** envset to be used by qremote
**QsgCmdProc[]** set qsg handler for suffix X
**QsgLibPath** search path for qsg libraries
**QsgLibs** additional qsg libraries and control flags
**QsgMap[script]** mapping for qsg script
**QvrsDir** directory corresponding to qvrs file being processed
**QvrsFile** name of the qvrs file being processed
**QvrsVersion** version number of the qvrs program
**RDIST** list of machines to which to rdist
**RdistFiles** list of files to be processed by rdist
**RealBranch** path from RootDir to the current directory
**ReportMkvernum** force mkvernum/vcc to report the version number
**Revision** revision number for the system
**RevisionString[X]** mkvernum format for ˆX
**Revision[X]** revision number for module
**RootAlias** alias or alternative name for the root directory
**RootDir** directory containing root.vrs file or the current directory
**RootPath** list of root directories
**RootVrsOwner** owner of the root.vrs file
**SLScmd** source database shell command
**SLSignores** list of patterns of files to be ignored by sls
**SOLIB_REVS** option that shared library revisions to be created
**STAND_ALONE** option indicating building stand-alone product
**SaveQefTrack** option to save the qef halt/track file
**SleepBeforeStat** sleep before stating a file may be required
**SrcPath** list of source directories
**SrcProdMap[class]** mapping of source suffix to product suffix for class
**SrcRoot** last directory of RootPath
**SrcRoots** list of root directories minus object root
**SubDir** path from current qvrs file to current directory
**Suffixes** list of suffixes matched by sls(1)
**SuppressExecution[backend]** back-end flag to suppress execution
**SysVrs** basename of qvrs file to be used as the sysvrs file
**SysVrsFile** pathname of the sysvrs file
**SysVrsPath** list of directories searched for @SysVrs file
**System** the system name
**SystemTargets** valid combinations of host and target cross compilations
**System[Base]** base type of the system
**System[CPU]** name of the machine or CPU
**System[Config]** name of the configuration
**System[Name]** short symbolic name for the system
**System[OS]** name of the operating system
**System[Release]** release of the system
**System[SysVrs]** basename of the sysvrs file to be used
**System[]** names and attributes for the system
**TargetSys** name of target host - defaults to BuildSys
**TouchPath** list of directories searched for _Touch_() files
**TreeList** list of parallel trees searched by qefdirs
**TreeType** type of the tree (e.g., source, object, ...)
**TreeType[X]** the TreeType of RootPath directory X
**TreedirsList** name of treedirs(1) input file list
**TreedirsPath** directories to be searched for treedirs input file
**UnpubDir** location of the unpub directory
**UpperCaseDepsSearches** specifies incls and libs to search for upper case
**UsePureLink** use purelink for all programs
**User** name of current user
**VCBranch** displacement from VCRoot of files for tree
**VCMap** the specification for mapping the repository to disk
**VCMap[X]** the specification for mapping the repository to disk
**VCRoot** root of the version tree if not LastRoot
**VCSys** the name of version system and the vci library
**VMS_DEST** indicates destination directory is a VMS file system
**WIN_APP** option specifying that a MS Windows/MFC app is being built
**WIN_ATX** option specifying that a MS ActiveX object is being built
**WIN_VSGEN_APP** option specifying that an MS App Wizard app is being built
**_<option>_*_[]** options for all or specific files
**_AsFlags_*_[]** optimize flags for all or specific files
**_CcFlags_*_[]** optimize flags for all or specific files
**_CxxFlags_*_[]** optimize flags for all or specific files
**_D_*[file]** −D and −U flags for program X
**_DefaultArgs_** default arguments to the back-end
**_DestDir_** the Destination directory
**_F_*[file]** flags for program X
**_F_cc[file.c]** flags to cc
**_F_cc_c[file.c]** flags for cc −c command
**_F_cc_o[file]** flags for cc −o command
**_F_instal[file]** extra flags used by instal(1)
**_F_mkvernum_cc[module]** cc flags used to compile mkvernum module
**_F_qlex[file.l]** flags for qlex
**_F_qyacc[file.y]** flags for qyacc
**_F_sls** extra flags for sls
**_Hostname_** the abbreviated name of the host
**_LdFlags_*_[]** optimize flags for all or specific files
**_NotNice_** the not nice option
**_Optimize_*_[]** optimize flags for all or specific files
**_ProjectVars_** name of project qef file to be included by vars.qh
**_Servers_[service]** names of host that provide specified service
**_Solib_value_** flags and controls for shared libraries
**_SuffixRules_[−defl]** default list of suffixes matched by sls(1)
**_SuffixRules_[−set]** set of suffix rules for sls(1)
**_T_*** name of program X
**_T_ar** full path of ar program
**_T_as** name of as program
**_T_binsh** pathname of the shell command interpreter
**_T_cc** name of the C compiler
**_T_cp** name of cp program
**_T_lex** name of lex binary to be used by qlex
**_T_ln** name of ln program
**_T_make** name of make program
**_T_mkdir** name of mkdir program
**_T_mv** name of mv program
**_T_postmail** name of mail program used to post mail
**_T_purify** name of the purify binary
**_T_pwd** name of pwd program
**_T_ranlib** name of ranlib program
**_T_rcp** name of rcp program
**_T_realcc** name of realcc program
**_T_regsvr32** name of regsvr32 program
**_T_rlogin** name of rlogin program
**_T_rm** name of rm program
**_T_rmdir** name of rmdir program
**_T_rsh** name of rsh (remote shell) program
**_T_sh** name of sh program
**_T_strip** name of strip program
**_T_yacc** name of yacc binary to be used by qyacc
**_Threads_*_[]** optimize flags for all or specific files

## Qvrs Functions: *functions(x-qvrs)

**_T_** function to return name of the argument tool
**basename** function to return basename of the argument

**dirname** function to return dirname of the argument
**dosfile** function to convert unix paths to dos paths
**empty** function to test if string is empty
**env** function to retrieve value of environment variable
**exists** function to test if file exists
**expr** function to convert expression to decimal string
**findfile** function to find 1st occurrence of file in path
**home** function to retrieve user's home directory
**insert** function to inserts words into string
**isdir** function to test if argument is an existing directory
**ixfile** function to convert dos paths to unix/interix paths
**join** function to join arbitrary strings into one
**maproot** function to map a root path according to user's mapping file
**match** function to test if any pattern matches string
**notempty** function to test if string is not empty
**option** function to test if option set
**paths** function to create pathnames of file
**prog** function to test if argument is an executable file in $PATH
**qvinfo** function to return current file name and line number
**qvset** function to test if a variable is defined
**resolve** function to convert pathname to absolute rooted form
**rootoftype** function to retrieve root of argument TreeType
**splitpath** function to split argument string at colons
**sys** function to test if @System[Name] is matched by patterns
**take1st** function to extract first word of string
**trait** function to retrieve trait(1) value
**value** function to retrieve variable's value or default
**varmatch** function to retrieve associative variable's value

## Qsg Keywords: ***qsg-kws(x-qsg)**

**!** run a shell command
**#** commenting lines
**<<** output range of lines
**<>** output a list with optional escape and indentation
**>** output a line
**Call** invoke a proc or another script
**abort** output diagnostic, call stack and die
**add** add elements to variable if not already members
**addfor** append new elements to for loop list
**append** append new elements to variable
**break** break out the enclosing loop
**call** call a proc or script
**close** close an open file
**continue** continue the enclosing loop
**cset** conditionally assign a list to a variable
**debug** turn the debugging dumps on or off
**drop1st** drop the first element of a variable
**dump** dump the call stack
**elif** conditional else
**else** else part of an if statement
**endfor** end of a for statement
**endproc** end of a proc
**endsummary** end of a summary
**endwhile** end of a while statement
**fatal** output diagnostic and abort
**fi** end of an if statement
**flush** flush an output file
**for** repeat statements once for each element of a list
**gappend** append to a global variable
**gcset** set a global variable, but only if unset or empty
**gprepend** prepend to a global variable
**gset** set a global variable
**gunset** unset a global variable
**if** the if statement
**interp** interpret the output of a shell command
**mapscript** map a script name to another script
**message** output a message to the diagnostic output
**noop** evaluate arguments but don't do anything
**nset** assign a numeric expression value to a variable
**open** open a file or pipeline for input or output
**prepend** prepend new elements to variable
**proc** define a proc
**prompt** output a prompt to the diagnostic output
**pushfid** open a file or pipeline pushing open fid onto lifo stack
**qsglib** load a qsg library
**rdecr** decrement a register
**remove** remove elements from a variable
**reopen** reopen a file or pipeline for input or output
**repeat** beginning of an until statement
**reset1st** reset a first time switch
**return** return from the current proc or script
**returnval** return from the current proc or script with a value
**rincr** increment named register
**rset** set a register to value
**set** assign a list to a variable
**set1st** set a first time switch
**setenv** set or unset an enviironment variable
**shell** run a shell command
**summary** specify and document flags for a script
**trace** turn call tracing on or off
**until** repeat statements until expression is non-zero
**while** while argument expression is non-zero, repeat nested statements
**write** write a string to an output file-descriptor

## Qsg Functions: ***functions(x-qsg)**

**#** function to cope with SCCS magic string
**1stset** function to test and set first-time-switch
**S** function to convert list to _S_() form
**access** function to test attributes of a file
**ateof** function to test if EOF encountered in an open file
**bustup** function to split value into list elements
**call** function to call a script or procedure
**caller** function to retrieve script name of caller
**callln** function to retrieve caller's line number
**callnm** function to retrieve name used to call the current script
**cgiform** function to process string returned by an HTML form
**check** function to check arithmetic expression syntax
**chk_msg** function to retrieve diagnostic from last check function
**chk_value** function to retrieve value from last check function
**date** function to format current time
**dosfile** function to convert unix paths to dos paths
**drop** function to drop specified elements from a list
**env** function to retrieve a shell environment variable
**exists** function to check the existence of a file
**exit** function to exit program
**expr** function to evaluate arithmetic expression
**exprlist** function to evaluate list of arithmetic expressions
**findfile** function to find a file in a src path
**flags** function to build value from flags
**global** function to retrieve global variable value
**gnames** function to return list of names of global variables
**gotscript** function to check if script accessible
**gpop** function to pop sublist from a global variable
**gvkeys** function to return keys for an associative array variable
**htmlencode** function to convert to html encoding
**index** function to select specified element from a list
**interp** function to interpret the output of a shell command
**isdir** function to check the existence of a directory
**isset** function to test first time switch
**ixfile** function to convert dos paths to unix/interix paths
**join** function to join arguments into single element list
**libraries** function to return list of loaded library symbols
**linenum** function to retrieve line number of last line read
**match** function to determine if a string matched by patterns
**mode** function to retrieve open fid's mode
**notset** function to test first time switch
**numf** function to format numbers
**option** function to test if qvrs option is set
**paths** function to create SrcPath names for a file
**pid** function to retrieve process number of current process
**qnames** function to return list of qvrs variables
**qvkeys** function to return keys for an associative array variable
**qvrs** function to retrieve value of qvrs variables
**qvrsexpr** function to retrieve qvrs expression value
**qvset** function to test if qvrs variable is set
**rand** function to returns a random number
**rdecr** function to decrement a register
**readline** function to read line from open file
**readstr** function to read a literal line from open file
**readword** function to read next word from open file
**retval** function to test if return value used
**reverse** function to reverse the order of a list
**rincr** function to increment a register
**rset** function to set a register
**rval** function to retrieve a register's value
**script** function to retrieve name of current script
**scriptnum** function to retrieve current script's number
**split** function to split arguments into a list
**status** function to return the exit status of shell command
**strcmp** function to compare two elements
**strlen** function to retrieve argument string's length
**subpath** function to return path minus specified leading directory
**sys** function to test qvrs @System[Name] value
**take** function to take first N elements of a list, dropping rest
**trait** function to retrieve value of traits variables
**uniq** function to retrieve unique sorted list of argument list
**varmatch** function to retrieve a qvrs associative variable's value

## Qsg TildeOps: ***tildeops(x-qsg)**

**~!=** @L~!=/str/ → elements of L that are not 'str'
**~!=h** @L~!=h/str/ → elements of L whose heads are not 'str'
**~!=t** @L~!=t/str/ → elements of L whose tails are not 'str'
**~!g** @L~!g/rxp/ → elements of L that do not match reg. exp.
**~!gh** @L~!gh/rxp/ → elements of L whose heads do not match reg. exp.
**~!gt** @L~!gt/rxp/ → elements of L whose tails do not match reg. exp.
**~!m** @L~!m/pat/ → elements of L that do not match 'pat'
**~!mh** @L~!mh/pat/ → elements of L whose heads do not match 'pat'
**~!mt** @L~!mt/pat/ → elements of L whose tails do not match 'pat'
**~!x** @L~!x/sfxs/ → elements of L whose extension is not in sfxs
**~#** @L~# → #th element of L
**~%** @L~% → L, with _%_ containing elements quoted
**~()** @L~(funct args ...) → return value of @(funct args ... @L)
**~-#** @L~-# → L minus the first # elements
**~.** @L~. → L, used to terminate list processing
**~:** @L~:/str/ → L, if not empty otherwise str
**~=** @L~=/str/ → elements of L that are 'str'
**~=h** @L~=h/str/ → elements of L whose heads are 'str'
**~=t** @L~=t/str/ → elements of L whose tails are 'str'
**~?** @L~?/S1/S2/ → L with non-zero (zero) elements replaced by S1 (S2)
**~A** @L~A → _A_(elements of L) – _%_ conversion
**~I** @L~I/str/ → index of str within L
**~S** @L~S → _S_(elements of L)
**~a** @L~a/str/ → L, with str appended to each element
**~d** @L~d → the directories of L
**~e** @L~e → the extensions (suffixes) of L
**~g** @L~g/rxp/ → elements of L that match reg. exp.
**~gh** @L~gh/rxp/ → elements of L whose heads match reg. exp.
**~gt** @L~gt/rxp/ → elements of L whose tails match reg. exp.
**~h** @L~h → the heads of L
**~j** @L~j → the elements of L joined together
**~l** @L~l → number of elements in L
**~m** @L~m/pat/ → elements of L that match shell pattern
**~mh** @L~mh/pat/ → elements of L whose heads match 'pat'
**~mt** @L~mt/pat/ → elements of L whose tails match 'pat'
**~n** @L~n → 1 if L is not empty, 0 otherwise
**~p** @L~p/str/ → L, with str prepended to each element
**~q** @L~q → L, with each element embedded in quotes
**~r** @L~r → the roots of L
**~s** @L~s/rxp/str/ → L, with 1st rxp matched substrings replaced by str
**~sg** @L~sg/rxp/str/ → L, with all rxp matched substrings replaced by str
**~t** @L~t → the tails of L
**~v** @L~v → 1 if L is void (i.e., empty), 0 otherwise
**~x** @L~x/sfxs/ → elements of L whose extension is in sfxs

## Qsg Standard Scripts: ***std_qsl(x-qsg)**

**arupdlib** create qef script to arupdate and install a library
**chknoopt** if Target suppressed by @NO_<opt> output warning recipe
**cmds_bp** create qef script to install bp (et al) and dirsetup files
**cmds_dat** create qef script to install data files
**cmds_man** create qef script to install manual sections
**cmds_sh** create qef script to install sh or csh scripts
**cmds_xdb** create qef script to process x_db source
**commands** create qef script to process and install practically anything
**compile** output shell command to compile object
**dashochk** check if −O flag redundant
**deflargs** set _DefaultArgs_ if not already set
**docinst** create qef script to install elimso'd document
**echo** echo arguments to stdout
**finstall** create qef script to install a file
**fsic** create qef/qsh file system integrity check script
**gencat** create qef script to gencat and install NLS files
**generant** generate a generic recipe and installs (replaces generic)
**generic** create qef generic script
**gotprog** subroutine that tests if argument file found in $PATH
**hanoi** run the towers of hanoi for N rings
**icomp** create qef script to create an InstallShield .z from a set of files.
**install** create qef script to install multiple files
**instdir** create qef script to mkdir directories
**instfls** create qef script to build instfls recipes
**item** output qef #item lines
**lastdone** create qef script to install release information file
**liblist** build and install library from object lists
**library** create qef script to create archive object library
**llib** create qef script to build llib-lX.ln for the argument files
**manlink** create qef script to create links in the man directories
**maxmkprocs** set and export MAXMKPROCS
**mimkvars** include mimkvars.qh if not previously included
**mkdeps** process dependency state files and install
**mkiscomp** create qef script to make an InstallShield .ins file from a .rul file
**mklib** create qef script to compile arguments and archive
**mklink** create qef script to create links
**mklock** create qef script mimk lock nodes
**mkobj** script used by others to output object construction recipe

**mkobj_asm** create qef script to make an object from an .asm file
**mkobj_c** create qef script to make an object for a .c file
**mkobj_et** create qef script to make an object for a .et file
**mkobj_l** create qef script to make an object for a .l file
**mkobj_mc** create qef script to generate .h and .rc for an object for a .mc file
**mkobj_o** create qef script to make an object for a .o file
**mkobj_rc** create qef script to make a resource object from a .rc file
**mkobj_s** create qef script to make an object for a .s file
**mkobj_src** create qef script to make a preprocessed .def file from a .src file
**mkobj_y** create qef script to make an object for a .y file
**mkobject** create qef script to convert source files to object files
**mkobjlist** create qef script to compile arguments and add name to ObjList._
**mkopts** create qef script to create an options header file
**mkosetup** function to set up mkobj_* compilation controls
**mkqsglib** create qef script to build qsg library
**mksolib** create a shared library
**nodeflt** prevent the default actions for the argument files
**nohist** create qef directives to suppress mimk history file
**proglib** create qef script to convert source files to a program via archive
**program** create qef script to convert argument files into a program
**pureeh** create qef script to create purify et al binaries
**qsgcmdproc** set up QsgCmdProc bindings
**qsl_init** std qsg library setup
**rcsupd** create qsh script to perform rcs merge functions
**rdist** create rdist script to distribute to target machines
**readlist** return list read from argument file or pipeline
**relinfo** create qef script to install release information file
**shcmd** run shell command
**shcmd** run shell command
**shfix** create qef script to shfix a file and install result
**solib** create qef script to create shared or dynamic library
**sort** return the sorted argument list
**srcsymln** create qef scripts to link source file to destination
**strfix** create qef script to configure and optionally install files
**striplib** create qef script to created stripped version of a library
**subdir** create qef script that invokes sub-directory qef
**target** output qef #target line consisting of target plus description
**touchdir** standard processing for a touchdir directory
**treesync** create qef script to synchronize the dest tree with the src tree(s)
**uilcomp** create qef script to make a Motif .uid file from a .uil file
**warnings** issue warnings about unused arguments
**wordlist** insert separators into lists and returns or outputs result
**yesno** prompt user for a yes or no response

## Qefpp Controls: *ctrls(x-qefpp)

**#append** append a string to a macro
**#assign** assign a value to a macro
**#comments** commenting mechanism
**#conddef** define a macro if not already defined
**#define** define a macro
**#donotremove** name files and directories not to be removed
**#elif** elif construct
**#else** else part of an #if statement
**#error** output an error message and die
**#export** export a shell environment variable
**#fappend** append a string to a file
**#fi** end of a #if construct
**#if** select or suppress lines by expression test
**#import** import a shell variable
**#include** include named file in processing
**#item** list target items supported by script
**#message** output a message
**#option** specify an option macro
**#popdef** pop pushed macro definition
**#pushdef** push a macro definition
**#readpipe** read a pipeline and process
**#return** return from current file
**#rm...** the remove file and directory directives
**#rminst** add files to removal list
**#rminstdir** add named trees to removal list
**#rminstpat** add matched files to removal list
**#rmobj** add files to removal list
**#rmobjdir** add named trees to removal list
**#rmobjpat** add matched files to removal list
**#servers** process _Servers_[service] to modify _T_service
**#setargs** set back-end arguments
**#shell** run a shell command
**#srcmap** set the source pathname for a file
**#target** describe targets supported by script
**#umask** specify umask setting
**#undef** undefine a macro
**#unport** undefine an environment variable

## Qefpp Macros: *macros(x-qefpp)

**Doto** qefdirs do up to control
**ForceExecution[backend]** back-end flag to force targets to be rebuilt
**Ito** qefdirs installed to control
**QTREE** the location of the <qtree-root>
**SuppressExecution[backend]** back-end flag to suppress execution
**_*Dir_** destination subdirectories
**_A_()** replaced by quoted _%_ converted pathname on the source path
**_Argc_()** number of back-end arguments
**_Argv_()** selected back-end arguments
**_AsFlags_()** the assembler flags
**_B_E_Arg_()** expression used to test for an application argument
**_B_E_Flags_** set of special application flags passed to children
**_BackEnd_** name of back-end processor
**_CcFlags_()** additional cc flags
**_CdMake_()** macro to do qef −d
**_Cwd_()** replaced by the current working directory
**_CxxFlags_()** additional C++ flags
**_D_<prog>** define flags for program <prog>
**_DefaultArgs_** default back-end arguments
**_Defined_()** expression used to test if macro defined
**_DestDir_** base of the destination tree
**_DosFile_()** convert unix path to dos path
**_Equal_()** expression used to test the equality of two strings
**_Exists_()** expression used to test existence of file
**_Expr_()** replace argument expression by decimal value
**_F_<prog>** flags for program <prog>
**_Hostname_** name of the host machine
**_IfOlder_()** file name, but only if older than rest of list
**_InclFlags_()** replaced by −IInclPath **...**
**_IsDir_()** expression used to test for a directory
**_IsMake_** macro defined if using make
**_IsMimk_** macro defined if using mimk
**_K_()** convert embedded _%_ strings to ' '
**_LdFlags_()** the loader flags
**_LdLibs_()** special case of _Libs_() for shared libraries
**_Libs_()** replaced by libraries for argument file
**_Literal_()** replaced by literal string
**_NotNice_** set to suppress nice()
**_NotNil_()** test if string is null
**_OnFailure_** command to be run after unsuccessful backend
**_OnSuccess_** command to be run after successful backend
**_Optimize_()** the optimization flags
**_PID_** process id of the qef process
**_PathSrc_()** replaced by a rooted pathname on the source path
**_Prog_()** expression used to test for an executable in the $PATH
**_ProjectVars_** name of project qef file to be included by vars.qh
**_QefFlags_** set of special qef flags passed to children
**_Quote_()** replaced by argument with embedded escapes for sh argument
**_QvrsExpr_()** macro used to retrieve qvrs functions and/or values
**_S_()** replaced by a pathname on the source path
**_Script_()** replaced by name of make script
**_ShVar_()** replaced by value of argument shell environment variable
**_SharedLibs_()** special case of _Libs_() for shared libraries
**_Solib_()** the shared library controls
**_StaticLibs_()** special case of _Libs_() for static libraries
**_Status_()** expression used to test status of last shell command
**_SuppressDeps_** boolean indicating that deps run is to be suppressed
**_T_<prog>** name of program <prog>
**_Threads_()** the thread flags
**_Touch_()** replaced by touch file if found
**_Umask_** required umask(2) setting
**_WalkSupported_** walk supported by script

**See also: refcard**