TURBO
QUICK REFERENCE GUIDE

by
Simon Goodwin

DIGITAL PRECISION

Published by Digital Precision, 222 The Avenue, London E4 9SE
Program and Documentation © 1986 Digital Precision

QL TURBO

Quick Reference Guide

29th May 2006

INTRODUCTION

This TURBO Quick Reference Guide contains two main areas

- Compiler and Runtime Error Messages
- TURBO TOOLKIT Commands

Both the Compiler and Runtime Error Messages, and the TURBO TOOLKIT Commands are listed alphabetically so that they can easily be referenced.

Section 1 - Compiler Messages

1.1 - PARSER ERROR MESSAGES

A FUNCTION MUST RETURN A VALUE

A Parser error message

Either you have tried to RETurn from a function without supplying a 'result' as a parameter of RETurn, or you have started a statement with the name of a function.

The second possibility indicates one of two logical errors. Either you've tried to call a function without specifying a destination for the value that will be returned, or you have tried to store a value in a variable that has the same name as a function.

AMBIGUOUS DECLARATION OF NAME

A Parser error message

A name declared on the line indicated has also been declared as an incompatible array, procedure, function, GLOBAL or EXTERNAL. This declaration makes the type of the name unclear.

AMBIGUOUS NAME USED

A Parser error message

The line shown contains a name which is of indeterminate type -perhaps it has been declared as an array and also as a function, for example.

This message may be generated more than once - this repetition is deliberate, so that you can easily find ALL the places where an ambiguous name is used.

ARRAY NAME REQUIRED

A Parser error message

The first parameter of DIMN should be the name of an array used in your program. This message may appear if there is no corresponding DIM for the name specified, or two DIMs with a different number of dimensions have been found for that name.

ARRAY OPERATION NOT ALLOWED

A Parser error message

The compiler has detected an attempt to 'slice' a numeric array, or otherwise manipulate a number of array elements within a single statement. Array slicing and manipulations are only allowed to act upon the last dimension of strings, unless the statement is DIM, LOCal, EXTERNAL, GLOBAL or REFERENCE, in which case the entire array should be specified.

Alternatively this message may indicate that you have tried to redimension an array which has been declared as EXTERNAL to this module. A shared array may only be redimensioned by the module in which it is GLOBAL (unless it is a parameter of a routine which is itself an EXTERNAL parameter).

ARRAYS ONLY PASSED BY REFERENCE

A Parser error message

You have placed parentheses around the name of an array. Remove them; there is no way to pass an array by value.

BUFFER TOO SMALL

A Parser error message

Your program contains a constant string (a text literal) which is larger than the 'buffer size' you set when you compiled the program. Since the minimum buffer size is 1024 characters, this is an extremely unlikely error! You can prevent the error by increasing the buffer size so that it exceeds the length of the longest sequence of characters enclosed by single or double quotation marks in your program.

CALCULATION NOT ALLOWED IN DATA

A Parser error message

Compiled data statements may not contain expressions. They may contain strings (enclosed in single or double quotation marks) and numeric constants, optionally preceded by a unary operator: NOT, "+", "-" or ~~ (bitwise negate).

CALCULATION TOO COMPLEX

A Parser error message

A name declared MANIFEST must be set equal to an explicit number or string.

Otherwise this message is extremely unlikely to appear unless you use a phenomenally complex expression - twenty brackets of the same type, a very long sequence of unary operators, or a lot of function calls one within another. You are too clever for your own good. You must simplify the expression or split it into several stages.

CAN'T ASSIGN VALUE - MANIFEST

A Parser error message

A name declared as MANIFEST cannot be treated as a variable and given a value.

CAN'T PASS EXPRESSION BY REFERENCE

A Parser error message

You have used a compound expression in calling a procedure or function whose DEFine statement is preceded by a REFERENCE statement for the corresponding parameter. Either assign the value of the expression to a variable which you then use in the routine call, or accept passing by value by deleting the offending parameter from the REFERENCE statement (or the whole REFERENCE statement if this was its only parameter).

CHANNEL NUMBER (0-31) NEEDED

A Parser error message

You must specify a valid channel number as the first parameter of this command. TURBO supports 32 channels, numbered from 0 to 31. If this error appears by a calculation which will always have a final value between 0 and 31, put brackets around the calculation to stop TURBO complaining.

COMPILATION ABORTED

A Parser error message

The circumstances described in the previous error report make it impossible for the compiler to continue scanning the program. Any subsequent errors are not detected.

CONFLICTING DIRECTIVE STATEMENTS

A Parser error message

You have used a name in both a GLOBAL and an EXTERNAL statement of the same module.

DEFINE / DIRECTIVE MISMATCH

A Parser error message

For a routine specification in a GLOBAL statement, there is no matching specification in a later DEFine statement, with the only difference being the omitted parentheses around any value parameter.

Another possibility is that a DEFine statement includes a name previously used in an EXTERNAL statement in the same module.

DUPLICATE GROUP NAME

A Parser error message

A group name occurs twice in non-consecutive GLOBAL or EXTERNAL statements of the same module.

END OF STATEMENT EXPECTED

A Parser error message

The compiler reached what it thought should be the end of a statement and then found extra characters there. This message can appear if you try to pass the wrong number of parameters to a procedure, or close more parentheses than you've opened, in an expression.

ERROR DIAGNOSIS FAILED

A Parser error message

An internal problem has occurred, causing the compiler to stop the compilation. Examine the listing at the point where the message was generated, to see if the cause is obvious. Otherwise there is a major fault in your computer or your copy of the compiler. Reset the QL and try again, in case the problem was caused by electrical interference.

FAULTY LINE REFERENCE

A Parser error message

The lines specified in RESTORE, GO TO, GO SUB and ON...GO statements must be fixed values, not the results of a calculation. Furthermore, they must be the numbers of lines that exist. Procedure names may be used instead of line numbers in ON...GO SUB, but the names must refer to BASIC procedures that expect no parameters, so that a 'GO SUB' is appropriate.

INCONSISTENT MODULE NUMBERS

A Parser error message

Either two GLOBAL statements have different module numbers, or an EXTERNAL statement has the same module number as the module's GLOBAL statement(s).

INCORRECT CALCULATION SYNTAX

A Parser error message

This message indicates that the compiler did not end up with a single value after evaluating an expression. Likely causes are unmatched or 'extra' brackets, or illegal characters in the expression.

INCORRECT LINK SPECIFICATION

A Parser error message

A GLOBAL or EXTERNAL statement has been found in the wrong part of a program, or there is an error in the format of the declaration. GLOBAL and EXTERNAL statements must come after all the DATA in a program, and after DEFinitions and DIMension statements which refer to their parameters.

INCORRECT NUMBER OF PARAMETERS

A Parser error message

A call to a SuperBASIC or machine-code routine has either too few or too many parameters.

INCORRECT SUPERBASIC SYNTAX

A Parser error message

A syntax error has been found. Normally these are detected by the SuperBASIC editor, which marks such lines with the keyword MISTake.

This message may also appear if the program being compiled is corrupt, or you have used the interpreter while TURBO's PARSER is running. The SuperBASIC data area should not be modified while TURBO runs, or the compiler may 'lose its place' and issue this message. The problem cannot occur if you invoke TURBO with the CHARGE command.

LOCALS MUST FOLLOW DEFINITIONS

A Parser error message

Local variables must be declared at the very beginning of a procedure or function - before other statements (apart from comments). This is a rule imposed by SuperBASIC.

LOOP DOES NOT EXIST HERE

A Parser error message

An EXIT or NEXT statement has been found after the END of the associated loop, or before the start of the loop.

Only one INS file allowed

A Parser error message

An attempt has been made to include more than one _INS file which contains a config block to be loaded.

ONLY SCALAR NAME FOR MANIFEST

A Parser error message

The name of a string declared MANIFEST must not have been dimensioned as a string array. (Dimensioning a name as a simple string before it is declared MANIFEST is allowed but gives rise to a Warning.)

PARSER DATASPACE FULL

A Parser error message

TURBO has not got enough space to keep details of all the line references, declarations and nested control constructs in your program. The simplest corrective actions are to increase the dataspace of PARSER_TASK by a few K, as explained in Section 2.4, or reduce the size of the program.

PITCH OUT OF RANGE: string

An error message of the PLAY_TUNE demo routine

The note shown is outside the 25-note range. Use X, or transpose the tune in a different key.

PROCEDURES DO NOT HAVE VALUES

A Parser error message

You're either trying to use a procedure name (which has no associated value) in IMPLICIT%, IMPLICIT$ or a calculation, or you're trying to RETurn a value when you're not in a function.

You should not use the RETURN command with a parameter (e.g. RETURN var) inside a procedure definition or at the end of a standard BASIC subroutine (the sort you'd call with GO SUB). Neither procedure calls nor GO SUB expect you to return a value.

If this report appears by a calculation, it generally means that the name has been mis-typed, or a procedure has been accidentally defined instead of a function.

ROUTINE NAMES CANNOT BE PARAMETERS

A Parser error message

You have accidentally put the name of a procedure or function in a REFERENCE directive. Variables of all types and shapes can be passed by REFERENCE, but not routines!

SIZE SHOULD BE BETWEEN 1K AND 850K! PLEASE TRY AGAIN

An error message of the DATASPACE_TASK program

You entered a request for dataspace that was out of bounds. Retry with a valid one.

STATEMENT IS NOT YET SUPPORTED

A Parser error message

The statement is in some QL ROMs but has not yet been documented as a formal part of SuperBASIC. This message may appear if INPUT, EOF or FOR are used in an invalid context.

This may also indicate an attempt to use a 'level' of hierarchical WHEN_ERROR trapping which is not currently supported.

VARIABLE ASSIGNMENT EXPECTED

A Parser error message

A reference to a variable was found at the start of a statement, yet it was not followed by an equals sign. This suggests that you may be trying to use a variable as if it were the name of a procedure.

WARNING: CLEAR DESTROYS ALL STRINGS

A Parser warning message

All strings in TURBO are arrays, even if they are 'automatically' dimensioned for you by TURBO. It follows that they are de-allocated by CLEAR.

If this warning appears on a CLEAR which is only executed right at the start of your program, your best bet is to remove the CLEAR, as TURBO automatically CLEARs variables at the start of a program.

If the CLEAR is executed while a program runs, you should make sure that you re-declare all the strings you will want to use later, particularly any which were automatically created at the start of the program by TURBO.

WARNING: Debug level not given - 0 assumed

A Parser warning message

A DEBUG command has been issued before DEBUG_LEVEL.

WARNING: Debug level restricted to 9

A Parser warning message

DEBUG_LEVEL has been given with a parameter greater than 9.

WARNING: Debug 0 assumed

A Parser warning message

A DEBUG command has been issued without a following integer.

WARNING: DEBUG restricted to 9

A Parser warning message

A DEBUG command has been issued with a following integer greater than 9.

WARNING: DIM not required for MANIFEST string

A Parser warning message

A name used for a string declared MANIFEST has been dimensioned. This is not necessary.

WARNING: DIM var$([40-257]) ASSUMED.

A Parser warning message

The string shown (var$) is used in your program but not dimensioned. TURBO has automatically dimensioned it to a maximum length of 100 characters.

WARNING: DIRECTIVE IN THE WRONG PLACE - IGNORED

A Parser warning message

TURBO has found a CONTINUE or RETRY statement outside a WHEN clause in a compiled program. If the interpreter found such an error, it would stop abruptly, with no message! TURBO is more polite and ignores the statement while warning you of its presence.

This warning may indicate that GLOBAL or EXTERNAL directives in your program have been ignored, because they were not at the start of the file. Move them by typing EDIT, changing the line number and deleting the old line. The auto-corrector indicates the relevant line numbers in its report. Then re-compile the program!

Finally, this warning appears if you try to use the SNOOZE or CATNAP directives in a program or module with no EXTERNAL routines in it.

WARNING: Duplicate Debug level ignored

A Parser warning message

A Debug level may only be specified once.

WARNING: END DEFINE sub ASSUMED

A Parser warning message

TURBO has encountered a DEFine or REFERENCE statement while it was still reading the definition of another procedure or function. The name of the definition being read appears in the message ('sub' in the above example). This warning means that an END DEFine is missing or in the wrong place.

The auto-corrector assumes that you meant to put an END DEFine before the next definition. This corrects the error, so long as you have not tried to nest one definition within another and there was not meant to be any code between the two definitions.

WARNING: END FOR ASSUMED

A Parser warning message

TURBO has reached the end of a block and realised that it has started a FOR loop in the meantime but never found the appropriate END. To keep things properly nested, the auto-corrector adds an END FOR here.

WARNING: END FOR var ASSUMED

A Parser warning message

TURBO has assumed an END FOR where it is required by the SuperBASIC syntax. In the example, 'var' corresponds to the name of the variable controlling the loop.

WARNING: END IF ASSUMED

A Parser warning message

TURBO has reached the end of a block and realised that it has started an IF since then, but never found the appropriate END. To keep things properly nested, the auto-corrector has added an END IF at this point.

WARNING: END REPEAT ASSUMED

A Parser warning message

TURBO has reached the end of a block and realised thatit has started a REPeat loop since then, but never found the appropriate END. To keep things properly nested, the auto-corrector adds an END REPeat here.

WARNING: END SELECT ASSUMED

A Parser warning message

TURBO has reached the end of a block and realised that it has started a SELect since then, but never found the appropriate END. To keep things properly nested, the auto-corrector adds an END SELect here.

WARNING: END TREATED AS NEXT

A Parser warning message

The TURBO auto-corrector has found a conditional END FOR. In correct SuperBASIC, END FOR is a 'static' marker, indicating the end of the scope of a loop. Every loop must have an END, and it is not sensible to have one that may be here one minute and gone the next!

The auto-corrector will issue a sequence of other messages to correct the error while preserving the behaviour of the program under the SuperBASIC interpreter. TURBO changes the compiled program, but leaves your SuperBASIC alone.

WARNING: END WHEN ASSUMED

A Parser warning message

TURBO has reached the end of a block and realised that it has started a WHEN_ERROR since then, but never found the appropriate END. To keep things properly nested, the auto-corrector adds an END_WHEN here.

WARNING: EXCESS PARAMETERS IGNORED

A Parser warning message

There are more parameters in the call than in the definition. The excess parameters (from the right) are ignored.

WARNING: EXIT var ASSUMED

A Parser warning message

TURBO is in the throes of correcting a conditional END FOR or END REPeat statement in your program. 'var' is the name of the loop identifier.

WARNING: Extensions are too long

A Parser warning message

The total length of the string which includes all the extension files exceeds 32766.

(This applies to Turbo v4.16 and later.)

WARNING: EXTRA ELSE CLAUSE IGNORED

A Parser warning message

TURBO has found a second (or subsequent!) ELSE matching a single IF statement. The extra ELSE, and all the code after it up to the END IF, is ignored.

WARNING: EXTRA END IGNORED

A Parser warning message

TURBO has found an END when it had not previously found a matching start of block. The extra END is ignored, just as it would be by the SuperBASIC interpreter.

WARNING: EXTRA END TREATED AS RETURN

A Parser warning message

TURBO has found an END when it is not scanning a definition. There are two possible reasons why this warning might appear: either one definition has been put inside another, which is not allowed and will cause other warnings; or one definition has more than one END. The auto-corrector has assumed the second case. Check the program to make sure that this correction is appropriate.

WARNING: Faulty extension string

A Parser warning message

A REMark %%.. string designed to cause inclusion of a file of SuperBASIC extensions is faulty.

(This applies to Turbo v4.16 and later.)

WARNING: File <file> can't be opened

A Parser warning message

The filename <file> requested as an extension can't be opened.

(This applies to Turbo v4.16 and later.)

WARNING: File <file> is too long

A Parser warning message

The filename <file> requested as an extension is longer than 32752.

(This applies to Turbo v4.16 and later.)

WARNING: LINE NOT FOUND num ASSUMED

A Parser warning message

The line referred to in the program does not exist. Like the interpreter, TURBO has assumed you meant to refer to the next existing line number - the one whose number is shown in the warning message.

WARNING: LOCAL var$([40-257]) ASSUMED

A Parser warning message

You have not specified an explicit maximum size for this local string, so TURBO has assumed a length of up to 100 characters.

WARNING: MEANINGLESS COMMAND IGNORED

A Parser warning message

The command indicated is incompatible with the form of a compiled program. When a TURBO task is executed, the original program text is not present. Thus commands such as LIST, MERGE, LRUN and so on are meaningless - they work with BASIC text, not compiled machine-code.

WARNING: MISSING PARAMETERS ASSUMED

A Parser warning message

There are more parameters in the definition than in the call. Zeroes and/or null strings have been added, as appropriate.

WARNING: NEXT TREATED AS END FOR

A Parser warning message

In a correct SuperBASIC program every FOR loop should end with an END FOR, just as every IF ends with an END IF and every SELect with an

END SELect. Short forms have an implied END at the end of the line concerned.

TURBO has detected an attempt to end a loop with NEXT rather than END FOR, as you would in 'traditional' BASIC. It assumes that you really meant END FOR and corrects the statement for you. You should really change the source, to avoid problems when testing the program with the interpreter.

WARNING: NUMERIC PARAMETER ASSUMED

A Parser warning message

TURBO cannot tell whether the name indicated in the report is that of a variable or a file or device. It has assumed that the name is that of a floating-point variable.

The variable will have its value passed to machine code, but subsequent changes will not affect its value.

The format of floating-point variable names and file names is identical in SuperBASIC, unless the name is put in inverted commas. If this message appears next to a filename, you should put the name in inverted commas to avoid ambiguity.

WARNING: var$ ONLY EXISTS AS A LOCAL

A Parser warning message

TURBO is warning you that the variable shown is a local or parameter string, but it is never dimensioned globally. Attempts to use it outside the procedures or functions where it is local will give a 'not found' error.

WARNING: REFERENCE TO name UNUSED

A Parser warning message

The variable name shown was amongst the latest group of REFERENCE names, yet it did not appear as a parameter in the subsequent DEFinition.

WARNING: Size optimised - Rest of line ignored.

A Parser warning message

A 'structured' program contains inadmissible instructions at the end of a line starting with DATA, DEBUG or DEBUG_LEVEL.

WARNING: Size optimised - line ignored.

A Parser warning message

A 'structured' program conatins an inadmisable line.

WARNING: SPARE END OR ELSE IGNORED

A Parser warning message

TURBO has found an END or ELSE when it had not previously found a matching start of block. The extra END or ELSE is ignored, just as it would be by the SuperBASIC interpreter.

WARNING: STOP ASSUMED

A Parser warning message

TURBO has reached the last statement of the main program, with only procedure and function definitions following. It assumes that you wish the program to stop when this point is reached and tells you so.

WARNING: Too many extensions

A Parser warning message

A call to include an extension file has been issued after eight have been accepted. Only eight are permitted.

(This applies to Turbo v4.16 and later.)

WARNING: Wrong extension file

A Parser warning message

A file requested as an extension file is too short to contain one or both of the initialisation routine or the definition block.

(This applies to Turbo v4.16 and later.)

WARNING: VALUE PARAMETER ASSUMED

A Parser warning message

The variable shown will have its value passed to machine code, but subsequent changes will not affect its value.

If this message appears, you should check that your program does not expect the value of the parameter indicated in the report to change as a result of the procedure or function call.

1.2 - CODE GENERATOR ERROR MESSAGES

CODE GENERATION FAILED, LINE number, REASON: NO PARSED CODE FOR ANALYSIS

A Code Generator error message

Code Generator cannot find intermediate code.

CODE GENERATION FAILED, LINE number, REASON: TOO LONG FOR <64K SETTING

A Code Generator error message

This error message can occur if the front panel code size setting is <64K but the object code size has just reached or exceeded 64K. The fix is to re-compile, either with the program size suitably reduced, or with the front panel setting for code size changed to >64K. Note that ways of reducing program size (other than the obvious way) include re-compiling with a BRIEF optimisation setting instead of FAST or REMs (or using REMs instead of FAST), reducing the number of channels, and opting to have line numbers excluded from the compiled program. Reducing the number of REMarks in the source program, or the level of indentation, or the length of variable names, or the amount of dataspace will have NO effect (none of the first three "get through" to the compiled task, and the fourth is an area separate from the code area and is not included in the 64K limit).

CODE GENERATION FAILED, LINE number, REASON: Faulty _INS file

A Code Generator error message

The file containing the config block to be loaded is faulty in some way. The solution is to recompile with the correct _INS file included.

1.3 - RUN-TIME ERROR MESSAGES

name IS NOT LOADED

A run-time error message (may also occur when the Parser is invoked)

A SuperBASIC extension which was used to compile the task that is to be executed is neither loaded into the QL's resident-procedure area nor found in any extensions included in the program. Save whatever is necessary, reset, load TURBO Toolkit (or other tooolkit appropriate to the task) and retry.

name IS REDEFINED

A run-time error message (may also occur when the Parser is invoked)

The name of a SuperBASIC extension which was used to compile the task that is to be executed is now in the QL's resident-procedure area under a different type (function instead of procedure or vice versa). Save whatever is necessary, reset, load TURBO Toolkit (or other tooolkit appropriate to the task) and retry.

TASK x HALTED, AFTER LINE num, ADJUST DATASPACE PLEASE

A run-time error message

The task fails because it has not been allocated enough space for data.

TASK x HALTED, AFTER LINE num, HARDWARE EXCEPTION num

A run-time error message

A processor-detected hardware error has occurred while a TURBO task is in control of the system (perhaps because of a misdirected POKE$ or overheating problem). Most tasks 'freeze' or crash the system after such errors - but TURBO tasks can cope with them, as long as they are detected by the processor or QL support chips.

If such a hardware error is detected, TURBO stops the task and attempts to empty buffers, close all channels, and de-allocate memory.

TASK x HALTED, AFTER LINE num, REASON: ALREADY EXISTS

A run-time error message

In general, this means that a file which was not supposed to exist has been found.

If this message appears when you try to invoke several modules with LINK_LOAD, it means that two or more of the files have the same module number. You must ensure that every module has a unique number, or other modules will not be able to tell the duplicates apart.

TASK x HALTED, AFTER LINE num, REASON: BAD OR CHANGED MEDIUM

A run-time error message (may also occur after code generation)

There's something wrong with the disk or cartridge the compiled program is using, or there has been an attempt to record on a write-protected microdrive cartridge, or the medium in the drive has been changed while the system was updating or writing to a file.

TASK x HALTED, AFTER LINE num, REASON: BAD PARAMETER

A run-time error message

This has its usual meaning - TURBO performs simple parameter checking when a program is compiled, but some mistakes can only be detected when a program is executed. 'Bad parameter' can be returned by any resident procedure or function if you supply too few, too many or inappropriate parameters.

TURBO itself generates this error if you try to PEEK or POKE a word or long word at an odd address, or if you supply a negative length to FILL$.

This message also crops up if you try to load an inappropriate task with LINK_LOAD, or if you try to EXEC a TURBO task that has been corrupted.

TASK x HALTED, AFTER LINE num, REASON: BUFFER FULL

A run-time error message

A line of input is too long for the area of memory that has been set aside to contain it. This may be because the end-of-line marker - CHR$(10) - is missing!

TURBO uses ALL the free memory within a task, up to a maximum of 32767 bytes, as a buffer for INPUT, so you can often avoid 'buffer full' errors by increasing the dataspace of a task.

TASK x HALTED, AFTER LINE num, REASON: CHANNEL NOT OPEN

A run-time error message (may also occur after code generation)

This message appears if you try to use a channel that is not open (surprise, surprise), or if you specify a channel number less than zero or greater than 31, or if you try to connect more than 32 pipes to a task. The last problem should never crop up if you load tasks with EXEC, EXECUTE or LINK_LOAD.

TASK x HALTED, AFTER LINE num, REASON: DRIVE FULL

A run-time error message (may also occur after code generation)

There's no room for any more data on a disk or microdrive cartridge. The file is automatically closed by TURBO when your task stops.

TASK x HALTED, AFTER LINE num, REASON: END OF FILE

A run-time error message

The program has tried to read from a pipe that has been closed and drained of data, or there's no more data in a file or available from a device, or you've tried to READ when there are no more DATA items in the compiled task. You can detect this situation before an error occurs with the EOF function.

TASK x HALTED, AFTER LINE num, REASON: ERROR IN EXPRESSION

A run-time error message

TURBO has been unable to convert a non-numeric string into a number. To see this error, write: X% = "NINE" ! To avoid this error in INPUT statements, use EDIT% / EDITF instead.

TASK x HALTED, AFTER LINE num, REASON: FORMAT FAILED

A run-time error message

There's no disk or tape in a drive, or the disk, tape, drive or interface is faulty or was removed during formatting, or the microdrive cartridge is write-protected.

TASK x HALTED, AFTER LINE num, REASON: IN USE

A run-time error message (may also occur after code generation)

The file or device is already being used, and no other task may use it until the one that got there first has finished.

TASK x HALTED, AFTER LINE num, REASON: INVALID JOB

A run-time error message

The program has referred to a task that doesn't exist.

TASK x HALTED, AFTER LINE num, REASON: NOT FOUND

A run-time error message (may also occur after code generation)

This generally indicates that a compiled program has tried to access an array - or a string - which does not exist, either because it is not LOCAL to the line shown, or has not been dimensioned, or has been destroyed by CLEAR. If the line shown uses strings, and your program uses CLEAR, you may find it helpful to recompile the program with the 'REPORT' strings option selected.

In file-handling commands, this error indicates that the QL cannot find a file or device which your program expects to exist. Use DEVICE_STATUS to trap this error.

This message also appears if your program tries to RETurn when it is not performing a subroutine, function or procedure call, or if you try to RETRY without having specified a return point with RETRY_HERE.

If this message appears as soon as you try to LINK_LOAD a group of files, it indicates that a module declared as EXTERNAL in one or more of the files is missing. Check that you have supplied all the required module names. If this doesn't cure the problem, check the EXTERNAL directives in the programs and make sure that corresponding GLOBAL declarations exist.

TASK x HALTED, AFTER LINE num, REASON: OUT OF MEMORY

A run-time error message

The most probable reason for this is a RESPR or ALLOCATION function call for which there is not enough memory left in RAM. Either make your program less greedy of space, or remove previous allocations or concurrent tasks.

TASK x HALTED, AFTER LINE num, REASON: OUT OF RANGE

A run-time error message

This indicates that your program has tried to refer to a non-existent array element or slice, or used an invalid colour number or a graphics co-ordinate outside the relevant window. It also appears if the value used in ON...GO TO or ON...GO SUB is less than one or more than the number of possible destinations.

TASK x HALTED, AFTER LINE num, REASON: OVERFLOW

A run-time error message

This message is generated if values go beyond the allowed range; for instance, if floating-point values exceed about 1.6E616 (which, to be fair, is an extremely big number!). (There is no check for floating-point 'underflow' - ever so extremely tiny numbers are treated as zero.) Division by zero is a favourite way of generating this error.

Overflow is reported if integers - including temporary results obtained in the course of a calculation - become less than -32768 or more than 32767. An overflow is also indicated if you try to concatenate two strings with a total length of more than 32767 characters.

TASK x HALTED, AFTER LINE num, REASON: READ ONLY

A run-time error message (may also occur after code generation)

A file exists and may be read but not written to, because the disk is write-protected or another program is already reading the file. Unfortunately the microdrive software in the QL does NOT issue this report if a cartridge is write-protected - it just gives a 'bad or changed medium' when it's bored with trying to write to the cartridge.

TASK x HALTED, AFTER LINE num, REASON: XMIT ERROR

A run-time error message

You've set the wrong 'parity' on something connected to one of the QL's 'ser' ports.

Section 2 - TURBO TOOLKIT Commands

ALLOCATION(bytes [,taskno%,tasktag%])

Reserves the specified number of bytes (rounded to an integer multiple of 8) in RAM (on the common heap) for use by the specified task; returns the address of the lowest byte reserved. If taskno% and tasktag% are not specified, the reservation will be made for the current task.

Cross-reference: DEALLOCATE

BASIC_ADR(num%)

Returns the address of the machine code function or procedure at the specified integer position in the interpreter's internal Name Table. If this is out of range, or if the name is not a machine code function or procedure, BASIC_ADR returns zero.

BASIC_B%(offset%)

Returns the value of the byte at the specified offset from the base of the SuperBASIC area in RAM.

Cross-reference: BASIC_L, BASIC_W%

BASIC_F(offset)

Returns the value of the six bytes starting at the specified offset from the current base of the SuperBASIC area in RAM, interpreted as a floating-point number.

Cross-reference: FLOAT$, PEEK_F

BASIC_INDEX%(name$)

Returns the entry number of the specified name in the SuperBASIC name table, if there is such an entry;

-12 if there is no such entry (the search is case-sensitive);

-7 if there is a mismatch between tables in the SuperBASIC interpreter.

Cross-reference: BASIC_NAME$

BASIC_L(offset)

Returns the value of the long word (4 bytes) at the specified even offset from the base of the SuperBASIC area in RAM.

Cross-reference: BASIC_B%, BASIC_POINTER, BASIC_W%

BASIC_NAME$(entry%)

Returns the identifier held under the specified entry number in the SuperBASIC name table.

If entry% is less than 0 or greater than the highest current entry number, a 'bad parameter' error is generated.

Cross-reference: BASIC_INDEX%, BASIC_TYPE%

BASIC_POINTER(offset)

Returns the address computed as the sum of the address of the base of the SuperBASIC area in RAM and the value of the long word (4 bytes) at the specified even offset from that base.

Cross-reference: BASIC_L

BASIC_TYPE%(entry%)

Returns a number indicating the type of the identifier held under the specified entry number in the SuperBASIC name table, as follows:

0 ... no type

1 ... string

2 ... floating point

4 ... integer

If entry% is less than 0 or greater than the highest current entry number, a 'bad parameter' error is generated.

Cross-reference: BASIC_NAME$

BASIC_W%(offset)

Returns the value of the word (2 bytes) at the specified even offset from the base of the SuperBASIC area in RAM.

Cross-reference: BASIC_B%, BASIC_L

CATNAP

If used in interpreted SuperBASIC: suspends SuperBASIC until the break key combination is entered; if used in TURBOcharged SuperBASIC:

suspends the current task - continuing to permit access to its routines by other tasks - until a calling task has returned from one of these routines.

Cross-reference: SNOOZE

CHANNEL_ID(#channel%)

Returns the QDOS identifier for the channel associated with the specified number in the current task. Errors: 'Channel not open'

Cross-reference: CONNECT, SET_CHANNEL

CHARGE [Taskname] | [/]

If used in interpreted SuperBASIC: executes (in EXECUTE_A mode) the tasks of the TURBO compiler, assigning the specified task name to the generated object program (unless changed by the user on the TURBO front screen). The task containing this command is suspended until the command has been executed or aborted. Execution is aborted with a 'not complete' error when the abort key combination (Alt + Space) is pressed. If used with "/", CHARGE starts compiling immediately if the settings are acceptable. If used in TURBOcharged SuperBASIC, CHARGE generates a warning message by the compiler and is otherwise ignored.

Cross-reference: DEFAULT_DEVICE, EXECUTE, EXECUTE_A

COMMAND_LINE

Makes window #0 the current input channel of the SuperBASIC interpreter.

Cross-reference: END_CMD, TYPE_IN

COMPILED

If used in interpreted SuperBASIC: returns 0; if used in compiled SuperBASIC: returns 1.

CONNECT [#]pipe_in% TO [#]pipe_out%

Opens a channel, to which the second number specified is assigned, as the output end of the pipe to whose input end the first channel number was assigned in an OPEN command.

Cross-reference: CHANNEL_ID, SET_CHANNEL

CURSOR_OFF [#window%]

If a cursor (flashing or not) is visible in the specified window, it is turned off, and the window can no longer be selected with the task-switching key combination. The default window for this command is #1.

Cross-reference: CURSOR_ON

CURSOR_ON [#window%[!]]

If a cursor is not visible in the specified window, it is made visible, and the window can be selected (making the cursor flash) with the task-switching key combination. If the exclamation mark is omitted, the window is automatically selected, and the cursor will flash.

Cross-reference: CURSOR_OFF

DATA_AREA kilobytes%

The same as TURBO_objdat

DATASPACE(filename$)

Returns the number of bytes reserved for the specified task file or one of the following error codes:

-2 ... the file is not a task file

-3 or -6 ... there is insufficient free memory to open a new channel to the device

-7 ... there is no device (or file) with that name

-9 ... the device is busy, or the file is being written to by another task

-12 ... the device is valid but the filename is not

-16 ... bad or changed medium

Cross-reference: FREE_MEMORY

DEBUG num%

No effect in SuperBASIC. Sets the following code at DEBUG level num% for a TURBO compiled program. See TurboS4.Txt.

Cross-reference: DEBUG_LEVEL

DEBUG_LEVEL num%

No effect in SuperBASIC. Sets the level at which marked sections of code will be compiled by TURBO. See TurboS4.Txt.

Cross-reference: DEBUG

DEALLOCATE address

Cancels the reservation of space on the common heap made with a call of the ALLOCATION function which has returned the specified address. Nothing is done if no such reservation has been made.

Cross-reference: ALLOCATION

DEFAULT_DEVICE devicename$

An unquoted string constant is accepted with or without the final underscore character. Makes the specified device the default device when no device name is included in the filename parameter of the CHARGE, EXECUTE, EXECUTE_A, EXECUTE_W, LINK_LOAD, LINK_LOAD_A, and LINK_LOAD_W commands. Has exactly the same effect as PROG_USE.

Cross-reference: CHARGE, EXECUTE, EXECUTE_A, EXECUTE_W, LINK_LOAD, LINK_LOAD_A, LINK_LOAD_W

DEVICE_SPACE([#]channel%)

Returns the number of unused bytes on the medium carrying the file referred to by the channel number. The channel number must refer to an open file on a directory device.

DEVICE_STATUS([access%,] filename$)

If the specified access:

0 ... OPEN

1 ... OPEN_IN

2 ... OPEN_NEW

-1 ... OPEN or OPEN_NEW

is possible, the function returns the number of free bytes on the device (or a spurious 3.355392E7 - roughly 2^25 - for non-directory devices). If the specified access is not possible, the function returns one of the following error codes:

-3 or -6 ... not enough memory to open a channel

-7 ... no such device/file

-9 ... device busy or file being written

-11 ... device full

-12 ... device valid, illegal filename/parameters

-16 ... bad or changed medium

-20 ... disk is write-protected or is being read

If no access type is given, 2 is assumed.

EDITF([#channel%,] default[$] [,maxlength%])

Asks for input in the specified channel (flashing cursor), offering the specified default value for editing, and returns the default (if a floating-point number) or the floating-point number keyed in and terminated with ENTER.

Cross-reference: EDIT%, EDIT$

EDIT%([#channel%,] default[$] [,maxlength%])

Asks for input in the specified channel (flashing cursor), offering the specified default value for editing, and returns the default (if an integer) or the integer keyed in and terminated with ENTER.

Cross-reference: EDITF, EDIT$

EDIT$([#channel%,] default[$] [,maxlength%])

Asks for input in the specified channel (flashing cursor), offering the specified default value for editing, and returns the default or the string keyed in and terminated with ENTER.

Cross-reference: EDITF, EDIT%

END_CMD

Closes the channel opened when a command file (a saved SuperBASIC program without line numbers) is MERGEd. It should therefore be the last command in such files. The command is ignored when entered via the command line.

Cross-reference: COMMAND_LINE, TYPE_IN

END_WHEN

If used in interpreted code: no action. If used in TURBOcharged code: marks the end of an error-trapping block. There must be exactly one END_WHEN command subsequent to each WHEN_ERROR command in a program.

Cross-reference: ERLIN%, ERNUM%, RETRY_HERE, WHEN_ERROR

ERLIN%

Returns the number of the last SuperBASIC line on which execution started before the last error. Zero is returned if:

- there has been no error;
- the last error occurred in a command line, in a command file, or a TURBOcharged program where the 'no line numbers' compiler option was selected.

Cross-reference: END_WHEN, ERNUM%, RETRY_HERE, WHEN_ERROR

ERNUM%

Returns the code number of the last error (see "Errors" in the Concepts Section of the QL manual). Zero is returned if there has been no error.

Cross-reference: END_WHEN, ERLIN%, RETRY_HERE, WHEN_ERROR

EXECUTE

EXECUTE [tn0$]

EXECUTE[#ch_in% TO | fname TO][tn0$][{TO tn$}][TO #ch_out%]

All tn..$ symbols stand for task names of the form

```
    [devicename$_]filename$ [! priority%[;optstring$]]
```

or

```
    [devicename$_]filename$  ;optstring$[!priority]
```

If one taskname is specified: executes the specified task at the specified priority and passes the specified command string to it. If several tasknames are specified: executes the tasks (with the first task owning all the others), setting up between any two adjacent files a pipe of 200 bytes length from channel #31 of the left-hand task (or from a data file or a specified SuperBASIC channel to be opened at the start) to channel #30 of the right-hand task (or a specified SuperBASIC channel to be opened at the end). Termination of the first tasks stops all others. Channels are not closed automatically.

Cross-reference: CHARGE, EXECUTE_A, EXECUTE_W, LINK_LOAD, LINK_LOAD_A, LINK_LOAD_W, OPTION_CMD$

EXECUTE_A

If used in interpreted SuperBASIC:

```
    EXECUTE_A [tn0$]
```

If used in TURBOcharged SuperBASIC:

```
    EXECUTE_A[#ch_in% TO | fname TO][tn0$][{TO tn$}][TO #ch_out%]
```

All tn..$ symbols stand for task names of the form

```
    [devicename$_]filename$ [! priority%[;optstring$]]
```

or

```
    [devicename$_]filename$  ;optstring$[!priority]
```

If one taskname is specified: executes the specified task at the specified priority and passes the specified command string to it. If several tasknames are specified: executes the tasks (with the first task owning all the others), setting up between any two adjacent files a pipe of 200 bytes length from channel #31 of the left-hand task (or from a data file or a specified SuperBASIC channel to be opened at the start) to channel #30 of the right-hand task (or a specified SuperBASIC channel to be opened at the end). Termination of the first tasks stops all others. Channels are not closed automatically. The task containing this command is suspended until the command has been executed or aborted.

Cross-reference: CHARGE, EXECUTE, EXECUTE_W, LINK_LOAD, LINK_LOAD_A, LINK_LOAD_W, OPTION_CMD$

EXECUTE_W

If used in interpreted SuperBASIC:

```
    EXECUTE_W [tn0$]
```

If used in TURBOcharged SuperBASIC:

```
    EXECUTE_W[#ch_in% TO | fname TO][tn0$][{TO tn$}][TO #ch_out%]
```

All tn..$ symbols stand for task names of the form

```
    [devicename$_]filename$ [! priority%[;optstring$]]
```

or

```
    [devicename$_]filename$  ;optstring$[!priority]
```

If one taskname is specified: executes the specified task at the specified priority and passes the specified command string to it. If several tasknames are specified: executes the tasks (with the first task owning all the others), setting up between any two adjacent files a pipe of 200 bytes length from channel #31 of the left-hand task (or from a data file or a specified SuperBASIC channel to be opened at the start) to channel #30 of the right-hand task (or a specified SuperBASIC channel to be opened at the end). Termination of the first task stops all others. Channels are not closed automatically. The task containing this command is suspended until the command has been executed.

Cross-reference: CHARGE, EXECUTE_A, EXECUTE_W, LINK_LOAD, LINK_LOAD_A, LINK_LOAD_W, OPTION_CMD$

EXTERNAL module_no% [,group_id$],{Var_spec|Array_spec|Routine_spec}

Where:

```
group_id$: a case-sensitive  string constant (!)
Var_spec: Float_name | Integer_name | String_name
Array_spec: Array_name({dummy_constant%})
Routine_spec: TURBO_F,|TURBO_P,Rout_name({Ref_para|Value_para})
Ref_para: Var_spec | Array_spec
Value_para: (Var_spec)
```

If used in interpreted SuperBASIC: is ignored. If used in TURBOcharged SuperBASIC: establishes (at link time) the correspondence between the variables, arrays, functions and/or procedures contained in the command and those specified in the GLOBAL declaration in the program module with the specified number (and the same group name, if any).

Cross-reference: GLOBAL, LINK_LOAD, LINK_LOAD_A, LINK_LOAD_W, REFERENCE, TURBO_F, TURBO_P

FLOAT$(number)

Returns a six-character string coding of the specified floating-point number.

Cross-reference: GETF, GET%, GET$, INPUT$, INTEGER$, STRINGF, STRING%, STRING$

FREE_MEMORY

If used in interpreted SuperBASIC: returns the number of bytes of unused contiguous memory in the system. If used in TURBOcharged SuperBASIC: returns the number of unused bytes in the dataspace of the current task.

Cross-reference: DATA_AREA, DATASPACE

FWINDOW%(#ch, xz, yz, xo, yo)

Function to help write programs which will work on a standard QL screen but which can also use high resolution screens. This function takes the same parameters as WINDOW though the channel parameter is not optional. Returns an error code if the window can't be redefined, otherwise redefines it just as WINDOW does and returns 0.

GET%([#]channel%)

A function to read binary integers from a channel. PRINT GET%(3) reads 2 bytes from channel three and returns a corresponding integer value. A hash before the channel number is optional.

Cross-reference: FLOAT$, GETF, GET$, INPUT$, INTEGER$, STRINGF, STRING%, STRING$

GETF([#]channel%)

Returns the floating-point number encoded in the six bytes starting at the file pointer of the specified channel and moves the file pointer beyond it. An 'overflow' error is generated if the six bytes do not correspond to a valid floating-point number.

Cross-reference: FLOAT$, GET%, GET$, INPUT$, INTEGER$, STRINGF, STRING%, STRING$

GET$([#]channel%)

Returns the string whose length is encoded in the two bytes starting at the file pointer of the specified channel and moves the file pointer beyond it (i.e. by 2 bytes + the number of bytes encoded in these 2 bytes).

Cross-reference: FLOAT$, GETF, GET%, INPUT$, INTEGER$, STRINGF, STRING%, STRING$

GetHEAD [#]channel%,address

Procedure to read a file header. Two parameters - a channel number, and the address of a 'buffer' - an area of memory where the file header can be stored, are required. Before using the command, the file must be opened in the normal way. To reserve the buffer - an area of 64 otherwise-unused bytes of memory - use ALLOCATION.

Cross-reference: SetHEAD

GLOBAL module_no% [,group_id$],{Var_spec|Array_spec|Routine_spec}

Where:

```
group_id$:    a case-sensitive string constant (!)
Var_spec:     Float_name | Integer_name | String_name
Array_spec:   Array_name({dummy_constant%})
Routine_spec: TURBO_F,|TURBO_P,Rout_name({Ref_para|Value_para})
Ref_para:     Var_spec | Array_spec
```

```
    Value_para:    (Var_spec)
```

If used in interpreted SuperBASIC: is ignored. If used in TURBOcharged SuperBASIC: establishes (at link time) the correspondence between the variables, arrays, functions and/or procedures contained in the command and those specified in the EXTERNAL declaration(s) with the specified number (and the same group name, if any) in other modules.

Cross-reference: EXTERNAL, LINK_LOAD, LINK_LOAD_A, LINK_LOAD_W, REFERENCE, TURBO_F, TURBO_P

IMPLICIT% {Variable}

If used in interpreted SuperBASIC code: no action. If used in TURBOcharged SuperBASIC code: assigns type integer to all specified variables though their names do not end in a percent sign.

Cross-reference: IMPLICIT$

IMPLICIT$ {Variable}

If used in interpreted SuperBASIC code: no action. If used in TURBOcharged SuperBASIC code: assigns type string to all specified variables though their names do not end in a dollar sign.

Cross-reference: IMPLICIT%

INPUT$([[#]channel%,]chars%)

Returns a string with a length of the specified number of characters as input from the specified channel. If input of the specified length is not available, execution will be suspended until it is. On a console channel no cursor will be shown. The default channel is #1.

Cross-reference: GETF, GET%, GET$

INTEGER$(number%)

Returns a two-character string coding of the specified integer.

Cross-reference: FLOAT$, GETF, GET%, GET$, INPUT$, STRINGF, STRING%, STRING$

LINK_LOAD task0$[!priority%],{task$[!priority%]}

task0$ and task$ stand for task names of the form:

```
    [devicename$_]filename$
```

1. Sets up each task with the specified priority, the first task owning all the others.
2. Builds a table pointing to each task, cross-referenced by the respective module numbers.
3. Makes each task allocate space and resolve addresses for variables, arrays and routines wholly contained in the module.
4. If any variable, array or routine is declared EXTERNAL in a task: suspends that task, obtains the addresses of the 'unresolved' items from the modules offering the corresponding GLOBALs, and patches the information into the suspended task, which is then released.
5. Starts each task at its first executable statement.

Cross-reference: CHARGE, EXECUTE, EXECUTE_A, EXECUTE_W, LINK_LOAD_A, LINK_LOAD_W

LINK_LOAD_A task0$[!priority%],{task$[!priority%]}

task0$ and task$ stand for task names of the form:

```
    [devicename$_]filename$
```

1. Sets up each task with the specified priority, the first task owning all the others.
2. Builds a table pointing to each task,cross-referenced by the respective module numbers.
3. Makes each task allocate space and resolve addresses for variables, arrays and routines wholly contained in the module.
4. If any variable, array or routine is declared EXTERNAL in a task: suspends that task, obtains the addresses of the 'unresolved' items from the modules offering the corresponding GLOBALs, and patches the information into the suspended task, which is then released.
5. Starts each task at its first executable statement.

The task containing this command is suspended until the command has been executed. Operation is halted on pressing certain keys (usually ALT/SPACE)

Cross-reference: CHARGE, EXECUTE, EXECUTE_A, EXECUTE_W, LINK_LOAD, LINK_LOAD_W

LINK_LOAD_W task0$[!priority%],{task$[!priority%]}

task0$ and task$ stand for task names of the form:

```
    [devicename$_]filename$
```

1. Sets up each task with the specified priority, the first task owning all the others.

2. Builds a table pointing to each task, cross-referenced by the module numbers.

3. Makes each task allocate space and resolve addresses for variables, arrays and routines wholly contained in the module.

4. If any variable, array or routine is declared EXTERNAL in a task: suspends that task, obtains the addresses of the 'unresolved' items from the modules offering the corresponding GLOBALs, and patches the information into the suspended task, which is then released.

5. Starts each task at its first executable statement.

The task containing this command is suspended until the command has been executed.

Cross-reference: CHARGE, EXECUTE, EXECUTE_A, EXECUTE_W, LINK_LOAD, LINK_LOAD_A

LIST_TASKS [[#]channel%]

For each current task on the QL, a line made up as follows is written to the specified channel:

```
taskname$,taskno%,tasktag%,priority%
```

If the priority is followed by the letter 's', the task is currently suspended; if by a full stop, the task is currently active. The default channel is #1.

Cross-reference: RELEASE_TASK, REMOVE_TASK, SET_PRIORITY, SUSPEND_TASK

LONGINTEGER(4_string$)

Converts any string of 4 bytes into a floating point number equal to the signed long integer value of the bytes. The string must be 4 bytes long or an error message will result.

Cross-reference: LONGINTEGER$

LONGINTEGER$(fp_no)

Function to convert any floating point number in the long integer range (-2,147,483,648 to 2,147,483,647) into a string of 4 bytes. This function and the previous one are handy for putting long integer values into strings or files when database programming.

Cross-refernece: LONGINTEGER

MANIFEST

Directive for TURBO. Ignored in SuperBASIC.

MOVE_MEMORY addr1 TO addr2, bytes

Copies the specified number of bytes from memory (starting at the first address) to RAM (starting at the second address). If the second address is less than 131072, a 'read only' error will be generated (only ROM there).

Cross-reference: SEARCH_MEMORY

OPTION_CMD$

If used in interpreted SuperBASIC: returns "". If used in TURBOcharged SuperBASIC: returns the option string passed to the current task in the EXECUTE, EXECUTE_A or EXECUTE_W command.

Cross-reference: EXECUTE, EXECUTE_A, EXECUTE_W

PEEK_F(address)

Returns the value of the six bytes starting at the specified address and interpreted as a floating-point number.

Cross-reference: BASIC_F, FLOAT$, POKE_F

PEEK$(address,bytes%)

Returns a string composed of the ASCII decoding of the specified number of bytes, starting at the specified address. An 'out of memory' error is returned if the specified number of bytes in RAM is not free.

Cross-reference: POKE$

POKE$ address, string$

Pokes the ASCII values of the characters composing string$ into RAM starting at the specified address.

Cross-reference: PEEK$

POKE_F adress,fp_no

Procedure pokes the 6-byte internal form of a floating point number into memory at the specified address. The address must be even.

Cross-reference: PEEK_F

POSITION[(#channel%)]

Returns the position of the file pointer (in bytes; start = 0) in the specified channel. The default channel is #1. A 'bad parameter' error is generated if the channel is not linked to an open file or pipe.

Cross-reference: SET_POSITION

REFERENCE {Variable|Array({dummy_subscript%})}

If used in interpreted SuperBASIC: no action. If used in TURBOcharged SuperBASIC: indicates that the specified variable(s)/array(s) are passed to the procedure or function whose definition follows by reference rather than by value. Any reference parameter in a preceding GLOBAL directive must be echoed exactly in the REFERENCE directive preceding the DEFine statement which corresponds to the GLOBAL directive's routine parameter containing the reference parameter.

Cross-reference: CATNAP, EXTERNAL, GLOBAL, LINK_LOAD, LINK_LOAD_A, LINK_LOAD_W, SNOOZE

RELEASE_TASK taskno%,tasktag%

If the specified task is active: no action. If the specified task is suspended: activates the task. An 'invalid Job' error is generated if the specification does not refer to an existing task.

Cross-reference: LIST_TASKS, REMOVE_TASK, SET_PRIORITY, SUSPEND_TASK

REMOVE_TASK taskno%,tasktag%

Interrupts the specified task (if executing) and removes it from the task table. If taskno% = -1, it refers to the current task, i.e. causes the current task to commit suicide. An 'invalid Job' error is generated if the specification does not refer to an existing task. A 'not complete' error is generated if removal of the SuperBASIC interpreter (task 0,0) is attempted.

Cross-reference: LIST_TASKS, RELEASE_TASK, SET_PRIORITY, SUSPEND_TASK

RETRY_HERE

If used in interpreted SuperBASIC: no action. In TURBOcharged code: Marks the point in the program to continue from after a RETRY statement has been executed inside a WHEN_ERROR loop.

Cross-reference: END_WHEN, WHEN_ERROR

SEARCH_MEMORY(address,bytes,string$)

Returns the memory address of the first occurrence of string$ between the specified address and (address+bytes-1), or zero if no match has been found.

Cross-reference: MOVE_MEMORY

SET_CHANNEL [#]channel%,channel_id

Closes the specified channel if open and opens a new channel with the specified number, assigning it the specified 32-bit QDOS channel identifier. If a channel with this identifier already exists in the same or in another task, it may henceforth also be accessed with the specified channel number from the calling task.

Cross-reference: CHANNEL_ID

SET_FONT [[#]channel%,] address1 [,address2]

Selects the font stored at address1 to be used henceforth as the first font (and optionally the font stored at address2 as the second font) for all output to the specified screen window. Address 0 selects the QL standard first or second font. The default channel is #1. The channel must be an open console window.

SetHEAD [#]channel,address

Procedure to write a file header. Two parameters - a channel number, and the address of a 'buffer' - an area of memory where the file header is stored, are required. To make the changes perminate, the file must be closed.

Cross-reference: SetHEAD

SET_POSITION #channel%, position

Sets the file pointer in the specified channel to the specified position (in bytes), or to the start or end of a file in case of a respectively too low and too high parameter value. The channel specification must refer to an open file.

Cross-reference: POSITION

SET_PRIORITY taskno%, tasktag%, priority%

Changes the priority of the specified task to the specified value. Legal priority values are 0 to 127. If jobno% is -1, the priority of the current task is affected. An 'invalid Job' error is generated if the specification does not refer to an existing task.

Cross-reference: LIST_TASKS, RELEASE_TASK, REMOVE_TASK, SUSPEND_TASK

SNOOZE

If used in interpreted SuperBASIC: suspends SuperBASIC until the break key combination is entered; if used in TURBOcharged SuperBASIC: suspends the current task indefinitely.

Cross-reference: CATNAP

STRINGF(text$)

Returns the floating-point number which would be encoded internally as the six-character string text$.

Cross-reference: FLOAT$, GETF, GET%, GET$, INPUT$, INTEGER$, STRING%, STRING$

STRING%(two_chars$)

Returns the numeric decoding of a two-character string, i.e. 256*CODE(char1$)+CODE(char2$) A 'bad parameter' error is generated if the string is not two characters long.

Cross-reference: FLOAT$, GETF, GET%, GET$, INPUT$, INTEGER$, STRINGF, STRING$

STRING$(2_plus_n_chars$)

Returns the decoding of: a string preceded by two characters encoding the string length. A 'bad parameter' error is generated if the string is not at least two characters long.

Cross-reference: FLOAT$, GETF, GET%, GET$, INPUT$, INTEGER$, STRINGF, STRING%

SUSPEND_TASK [taskno%,tasktag%,] frames%

Suspends execution of the specified task for the specified number of fiftieths (for 50 Hz power supply) or sixtieths (for 60 Hz power supply) of a second or (if frames% is -1) indefinitely. The suspension may be lifted prematurely by RELEASE_TASK called in another task or (in the case of the SuperBASIC interpreter only) by the Break keypress combination. If the task is already suspended, the suspension period is reset from the moment of execution. If no task is specified or taskno% = -1, the current task is suspended. An 'invalid Job' error is generated if the specification does not refer to an existing task.

Cross-reference: LIST_TASKS, RELEASE_TASK, REMOVE_TASK, SET_PRIORITY

SYS_VARS

Function that returns the addres of System Variables.

TURBO_DUMMY%

A function which returns the error 'not implemented' in SuperBASIC. In a TURBO compiled program it reserves space in the Vector Table

Cross-reference: TURBO_DUMMY$, TURBO_DUMMYF, TURBO_DUMMYP

TURBO_DUMMY$

A function which returns the error 'not implemented' in SuperBASIC. In a TURBO compiled program it reserves space in the Vector Table

Cross-reference: TURBO_DUMMY%, TURBO_DUMMYF, TURBO_DUMMYP

TURBO_DUMMYF

A function which returns the error 'not implemented' in SuperBASIC. In a TURBO compiled program it reserves space in the Vector Table

Cross-reference: TURBO_DUMMY%, TURBO_DUMMY$, TURBO_DUMMYP

TURBO_DUMMYP

A procedure which returns the error 'not implemented' in SuperBASIC. In a TURBO compiled program it reserves space in the Vector Table

Cross-reference: TURBO_DUMMY%, TURBO_DUMMY$, TURBO_DUMMYF

TURBO_F

This keyword must only be used in TURBOcharged SuperBASIC, where, if used in an executable statement, it would execute exactly like FREE_MEMORY. Its real use, however, is within the EXTERNAL and GLOBAL directives, which see.

TURBO_P

This keyword must only be used in TURBOcharged SuperBASIC, where, if used in an executable statement, it would execute exactly like FREE_MEMORY. Its real use, however, is within the EXTERNAL and GLOBAL directives, which see.

TURBO_V(name$)

In SuperBASIC this function returns 1. In a TURBO compiled program it returns the absolute address in the Vector Table of name$.

TURBO_buffersz

Compiler directive to set the buffer space to be used by TURBO when compiling the program.

Cross-reference: TURBO_diags, TURBO_list, TURBO_locstr, TURBO_model, TURBO_objdat, TURBO_objfil, TURBO_objstk, TURBO_optim, TURBO_ref, TURBO_repfil, TURBO_sound, TURBO_struct, TURBO_taskn, TURBO_windo

TURBO_diags

Compiler directive to exclude, display or include the line-number diagnostic information in the compiled program. 0 excludes it for a smaller file, 1 displays only and 2 includes it; also uses strings like "I" or "INCLUDE" to include or "D" or DISPLAY" to display but not include, or "O" or "OMIT" to omit the information.

Cross-reference: TURBO_buffersz, TURBO_list, TURBO_locstr, TURBO_model, TURBO_objdat, TURBO_objfil, TURBO_objstk, TURBO_optim, TURBO_ref, TURBO_repfil, TURBO_sound, TURBO_struct, TURBO_taskn, TURBO_windo

TURBO_list

Compiler directive that uses 0, 1, "NO" or "YES to determine whether listing is done or not. The numbers 0 and 1 are alternatives to "NO" and "YES"

Cross-reference: TURBO_buffersz, TURBO_diags, TURBO_locstr, TURBO_model, TURBO_objdat, TURBO_objfil, TURBO_objstk, TURBO_optim, TURBO_ref, TURBO_repfil, TURBO_sound, TURBO_struct, TURBO_taskn, TURBO_windo

TURBO_locstr

Compiler directive that uses "I" or "IGNORE", "R" or "REPORT", "C" or "CREATE" to set the option for treatment of local strings; see TURBO manual for more details of the meaning of this. Also accepts 0, 1 or 2 for Ignore, Report or Create respectively.

Cross-reference: TURBO_buffersz, TURBO_diags, TURBO_list, TURBO_model, TURBO_objdat, TURBO_objfil, TURBO_objstk, TURBO_optim, TURBO_ref, TURBO_repfil, TURBO_sound, TURBO_struct, TURBO_taskn, TURBO_windo

TURBO_model

Compiler directive that selects 16-bit or 32-bit code production mode, use "<" for less than 64kb of code (16-bit) or ">" for more than 64kb programs, (which is 32 bit). Does not affect the amount of dataspace allowed, just the maximum size of executeable file produced. Also accepts numbers, 0 selects 16-bit code, 1 selects 32-bit code.

Cross-reference: TURBO_buffersz, TURBO_diags, TURBO_list, TURBO_locstr, TURBO_objdat, TURBO_objfil, TURBO_objstk, TURBO_optim, TURBO_ref, TURBO_repfil, TURBO_sound, TURBO_struct, TURBO_taskn, TURBO_windo

TURBO_objdat

Compiler directive that sets the number of kilobytes of dataspace to give the compiled program, so is equivalent to DATA_AREA.

Cross-reference: TURBO_buffersz, TURBO_diags, TURBO_list, TURBO_locstr, TURBO_model, TURBO_objfil, TURBO_objstk, TURBO_optim, TURBO_ref, TURBO_repfil, TURBO_sound, TURBO_struct, TURBO_taskn, TURBO_windo

TURBO_objfil

Compiler directive that sets the output filename for the compiled program.

Cross-reference: TURBO_buffersz, TURBO_diags, TURBO_list, TURBO_locstr, TURBO_model, TURBO_objdat, TURBO_objstk, TURBO_optim, TURBO_ref, TURBO_repfil, TURBO_sound, TURBO_struct, TURBO_taskn, TURBO_windo

TURBO_objstk

Compiler directive that sets the number of bytes of stackspace for the compiled task.

Cross-reference: TURBO_buffersz, TURBO_diags, TURBO_list, TURBO_locstr, TURBO_model, TURBO_objdat, TURBO_objfil, TURBO_optim, TURBO_ref, TURBO_repfil, TURBO_sound, TURBO_struct, TURBO_taskn, TURBO_windo

TURBO_optim

Compiler directive that sets optimisation type to be used, e.g. "B" or "BRIEF" for smallest code, "R" or "REMS" for switchable control by <REMark +> and <REMark -> statements and "F" or "FAST" for fastest (but very bulky) code. Will use 0 to 2 instead but strings are clearer.

Cross-reference: TURBO_buffersz, TURBO_diags, TURBO_list, TURBO_locstr, TURBO_model, TURBO_objdat, TURBO_objfil, TURBO_objstk, TURBO_ref, TURBO_repfil, TURBO_sound, TURBO_struct, TURBO_taskn, TURBO_windo

TURBO_ref

Compiler directive that sets simple string parameters to machine code routines by reference (SMSQ only).

Cross-reference: TURBO_buffersz, TURBO_diags, TURBO_list, TURBO_locstr, TURBO_model, TURBO_objdat, TURBO_objfil, TURBO_objstk, TURBO_optim, TURBO_repfil, TURBO_sound, TURBO_struct, TURBO_taskn, TURBO_windo

TURBO_repfil

Compiler directive that sets the report filename; a null string selects the display for the report.

Cross-reference: TURBO_buffersz, TURBO_diags, TURBO_list, TURBO_locstr, TURBO_model, TURBO_objdat, TURBO_objfil, TURBO_objstk, TURBO_optim, TURBO_ref, TURBO_sound, TURBO_struct, TURBO_taskn, TURBO_windo

TURBO_sound

Compiler directive that sets sound off or on. "NO" or 0 sets it off; "YES" or 1 sets it on.

Cross-reference: TURBO_buffersz, TURBO_diags, TURBO_list, TURBO_locstr, TURBO_model, TURBO_objdat, TURBO_objfil, TURBO_objstk, TURBO_optim, TURBO_ref, TURBO_repfil, TURBO_struct, TURBO_taskn, TURBO_windo

TURBO_struct

Compiler directive Instructs Turbo whether to treat the program as structured, i.e. structured is denoted by 1, unstructured by 0. Use "S" or "STRUCTURED" or "F" or "FREEFORM" instead of 0 or 1 for clarity. See Turbo manual for definitions of structured and freeform programs.

Cross-reference: TURBO_buffersz, TURBO_diags, TURBO_list, TURBO_locstr, TURBO_model, TURBO_objdat, TURBO_objfil, TURBO_objstk, TURBO_optim, TURBO_ref, TURBO_repfil, TURBO_sound, TURBO_taskn, TURBO_windo

TURBO_taskn

Compiler directive that sets the job name of the compiled program as seen by JOBS and similar commands, to a maximum length of 12 characters; anything longer is truncated rather than rejected.

Cross-reference: TURBO_buffersz, TURBO_diags, TURBO_list, TURBO_locstr, TURBO_model, TURBO_objdat, TURBO_objfil, TURBO_objstk, TURBO_optim, TURBO_ref, TURBO_repfil, TURBO_sound, TURBO_struct, TURBO_windo

TURBO_windo

Compiler directive Sets the number of automatic window definitions to copy from the SuperBASIC interpreter windows. Must be in the range 0-31.

Cross-reference: TURBO_buffersz, TURBO_diags, TURBO_list, TURBO_locstr, TURBO_model, TURBO_objdat, TURBO_objfil, TURBO_objstk, TURBO_optim, TURBO_ref, TURBO_repfil, TURBO_sound, TURBO_struct, TURBO_taskn

TYPE_IN text$

Causes the specified text to be entered on the SuperBASIC interpreter's command line (and executed, if the text terminates with the ENTER character CHR$(10))

Cross-reference: COMMAND_LINE, END_CMD

TK_VER$

Returns the current version number of TURBO Toolkit.

WHEN_ERROR [0|1]

If used in interpreted SuperBASIC: no action. If used in TURBOcharged SuperBASIC: skips to the associated END_WHEN statement but marks the start of a routine to be executed when an error occurs. If the error occurs within a WHEN_ERROR 1 routine and a WHEN_ERROR 0 statement has been executed, the WHEN_ERROR 0 routine is executed.

Cross-reference: END_WHEN, ERLIN%, ERNUM%, RETRY_HERE

Table of Contents