TurboForth Language Reference Manual

TurboForth

Category 1

TurboForth Kernal Glossary

The TurboForth Kernal glossary is divided into the following 20 categories:

Type	Synopsis
Block I/O Words	Words associated with performing disk block I/O.
Built-in Constants	Lists the built in constants.
Built-in Variables	Lists the built in variables.
Comparison Words	Words that allow comparisons between numbers.
Compilation Words	Words associated with compiling word references or data to memory.
Console I/O Words	Words to allow data to be written to the screen, keyboard scanning etc.
File I/O Words	Words associated with performing disk file I/O.
Flow Control Words	Words that allow the execution flow of a program to be altered.
Graphics Words	Words associated with graphics, such as colour, sprites, etc.
Interpreter Words	Words associated with the TurboForth interpreter/compiler.
Logical Words	Words associated with logical (bit wise) functions, such as AND OR NOT etc.
Math Words	Words directly associated with mathematical functions.
Memory Access Words	Words that allow memory to read from and written to.
Miscellaneous	This section details various misceallaneous words that cannot be categorised elsewhere.
Parsing Words	Words associated with extracting words from the input stream, dictionary look-up etc.
Return Stack Words	This section lists the words that manipulate the return stack
Sound Words	Words associated with generating sound.
Speech Words	Words associated with generating speech.
Stack Words	This section lists the words that manipulate the data stack.
String Words	Words associated with creating and manipulating strings.

Words in category Block I/O Words in glossary TurboForth Kernal

The following words are listed in this category (22 words found): **Word Types:** S = Standard, I = Immediate , IC = Immediate-compiling.

Word Name	Word Type	Data Stack Signature	Return Stack Signature	Description	Availability V1.0	V1.1	V1.2
-->	I	--	--	Loads the next block from the file currently in USE.	☑	☑	☑
BLK	S	-- address	--	Contains the number of the block currently being accessed, or 0 if no block is currently being accessed.	☑	☑	☑
BLK?	S	buffer -- block vdp_addr	--	Returns the disk block stored in the buffer *buffer* (or 0 if the buffer is empty) and the VDP address *vdp_addr* of the buffer.	☑	☑	☑
BLOAD	S	block -- nfb	--	Loads a block of raw memory from block storage that has been previously saved with BSAVE.	☑	☑	☑
BLOCK	S	block -- vdp_addr	--	Loads block *block* from the file currently in USE into a free buffer.	☑	☑	☑
BSAVE	S	start block -- nfb	--	Saves a block of memory to the current blocks file.	☑	☑	☑
BUF?	S	block -- buffer vdp_addr	--	Returns the *buffer* number of block *block*, and its address in VDP memory as *vdp_addr*. Returns 0 0 if the block is not in memory.	☑	☑	☑
CLEAN	S	buffer --	--	Sets the status of buffer *buffer* to clean. Subsequent FLUSHes will not flush the buffer to disk.	☑	☑	☑
CLOAD	I	block --	--	Conditionally loads *block* if the word immediately following CLOAD is not found after a dictionary search.	☑	☑	☑
DIRTY	S	buffer --	--	Sets buffer *buffer*'s status to dirty. Subsequent FLUSHes will causes the buffer to be written to disk, to the block set with SETBLK.	☑	☑	☑
DIRTY?	S	buffer -- flag	--	Returns TRUE if buffer *buffer* is dirty (i.e.the buffer has changed since being loaded from disk (via BLOCK or LOAD and requires flushing to disk), otherwise returns FALSE.	☑	☑	☑
EMPTY-BUFFERS	S	--	--	Unconditionally sets all buffers to CLEAN. Subsequent FLUSHes will not write anything to disk.	☑	☑	☑
FLUSH	S	--	--	Flushes all dirty buffers back to their respective block positions on disk. Buffers not marked as dirty are not flushed.	☑	☑	☑
LIST	S	block --	--	Displays block *block* to the screen.	☑	☑	☑
LOAD	S	block --	--	Loads (compiles/executes) block block.	☑	☑	☑
MKBLK	S	size --	--	Creates a block file (used for storing source code and/or data) on the nominated disk, with enough space for size screens/blocks of data and sets the new file as the current blocks file.	☐	☐	☑
MKDSK	I	size --	--	Creates a block file with the name file and capacity for *size* blocks.	☑	☑	☑
SETBLK	S	buffer block --	--	Associates buffer *buffer* with block *block*.	☑	☑	☑
THRU	S	start end --	--	Loads blocks starting at block *start* thru *end* inclusive.	☑	☑	☑
UPDATE	S	--	--	Marks the last accessed block as dirty.	☑	☑	☑
USE	S	string --	--	Defines the block file to be used. A string should be set up on the stack before calling USE.	☑	☑	☑
WHERE	S	-- block	--	Returns the block number of a user-loaded word in the dictionary. Non-existant words return 0.	☑	☑	☑

-->

In Block I/O Words in TurboForth Kernal

Word Name:	-->
Type:	Immediate word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Loads the next block from the file currently in USE.
Example:	none
Comment:	none
See Also:	THRU USE

BLK

In Block I/O Words in TurboForth Kernal

Word Name:	BLK
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Contains the number of the block currently being accessed, or 0 if no block is currently being accessed.
Example:	none
Comment:	Note: BLK is a variable (it returns its address) and should be accessed via @. Writing to BLK is not generally meaningful, it is managed internally by code within BLOCK.
See Also:	BLOCK

BLK?

In Block I/O Words in TurboForth Kernal

Word Name:	BLK?
Type:	Standard word
Data Stack Signature:	buffer -- block vdp_addr
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns the disk block stored in the buffer *buffer* (or 0 if the buffer is empty) and the VDP address *vdp_addr* of the buffer.
Example:	none
Comment:	none
See Also:	BUF? CLEAN DIRTY DIRTY? SETBLK

In Block I/O Words in TurboForth Kernal

BLOAD

In Block I/O Words in TurboForth Kernal

Word Name:	BLOAD
Type:	Standard word
Data Stack Signature:	block -- nfb
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Loads a block of raw memory from block storage that has been previously saved with BSAVE.
Example:	none
Comment:	BLOAD checks word 0 of the *block* for >994A. If >994A is not found then the data in the block is determined to be of invalid format and the operation fails, passing *block* to the stack. In other words, if after executing BLOAD the value on the stack is equal to *block*, the operation failed. BLOAD uses the header in each block and installs the payload contents (1008 bytes) at the address defined by the offset + 6 (see BSAVE for offsets). BLOAD continues loading blocks until it encounters a block that does not have >994A in offset + 0. Upon completion, BLOAD pushes the next free block number (*nfb*) to the stack (i.e. the last BLOADed block+1). It is not neccessary to understand the inner workings/format of the BLOAD/BSAVE block format. BSAVE takes care of formatting the data correctly in a form that BLOAD expects. **Note:** A program saved as an image with BSAVE can only be executed on a TurboForth system of the same version. In other words, it is not possible to BSAVE a program in V1.1 and execute it in version 1.2. The program should be re-compiled from the source code, and BSAVEd on the version in which it is to be executed.
See Also:	BSAVE

In Block I/O Words in TurboForth Kernal

Word Name:	BSAVE
Type:	Standard word
Data Stack Signature:	start block -- nfb
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Saves a block of memory to the current blocks file.
Example:	none
Comment:	Saves CPU memory from *start* to HERE to the block beginning at *block* and continuing for as many blocks as are required to hold the contents of memory. Upon completion, the stack contains the next free block number (*nfb*) (i.e. the last block used by BSAVE+1). BSAVEd blocks have the following format:
See Also:	LATEST HERE BLOAD

Offset	Description
0	>994a - marker for BLOAD
2	LATEST - current value of LATEST
4	HERE - current value of HERE
6	destination address of payload data
8	reserved for future use
10	reserved for future use
12	reserved for future use
14	reserved for future use
16 - 1023	1008 bytes of payload data

BUF?

In Block I/O Words in TurboForth Kernal

Word Name:	BUF?
Type:	Standard word
Data Stack Signature:	block -- buffer vdp_addr
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns the *buffer* number of block *block*, and its address in VDP memory as *vdp_addr*. Returns 0 0 if the block is not in memory.
Example:	none
Comment:	none
See Also:	BLK? CLEAN DIRTY DIRTY? SETBLK

CLEAN

In Block I/O Words in TurboForth Kernal

Word Name:	CLEAN
Type:	Standard word
Data Stack Signature:	buffer --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Sets the status of buffer *buffer* to clean. Subsequent FLUSHes will not flush the buffer to disk.
Example:	none
Comment:	none
See Also:	DIRTY DIRTY? FLUSH

CLOAD

In Block I/O Words in TurboForth Kernal

Word Name:	CLOAD
Type:	Immediate word
Data Stack Signature:	block --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Conditionally loads *block* if the word immediately following `CLOAD` is not found after a dictionary search.
Example:	`69 CLOAD SAMS?` The above will load block 69 if the word SAMS? is *not* found in the dictionary.
Comment:	Note, this is an immediate word. Not to be used in a colon definition. Typically, this word will only be used inside of a block, outside of a colon definition, to load an extension library if the library is not already loaded.
See Also:	LOAD

DIRTY

In Block I/O Words in TurboForth Kernal

Word Name:	DIRTY
Type:	Standard word
Data Stack Signature:	buffer --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Sets buffer *buffer*'s status to dirty. Subsequent `FLUSH`es will causes the buffer to be written to disk, to the block set with `SETBLK`.
Example:	none
Comment:	none
See Also:	BLK? BUF? DIRTY DIRTY? SETBLK

DIRTY?

In Block I/O Words in TurboForth Kernal

Word Name:	DIRTY?
Type:	Standard word
Data Stack Signature:	buffer -- flag
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns TRUE if buffer *buffer* is dirty (i.e.the buffer has changed since being loaded from disk (via BLOCK or LOAD and requires flushing to disk), otherwise returns FALSE.
Example:	none
Comment:	none
See Also:	BLK? BLOCK BUF? CLEAN DIRTY DIRTY? FLUSH LOAD

EMPTY-BUFFERS

In Block I/O Words in TurboForth Kernal

Word Name:	EMPTY-BUFFERS
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Unconditionally sets all buffers to CLEAN. Subsequent FLUSHes will not write anything to disk.
Example:	none
Comment:	none
See Also:	CLEAN FLUSH

FLUSH

In Block I/O Words in TurboForth Kernal

Word Name:	FLUSH
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Flushes all dirty buffers back to their respective block positions on disk. Buffers not marked as dirty are not flushed.
Example:	none
Comment:	none
See Also:	EMPTY-BUFFERS

LIST

In Block I/O Words in TurboForth Kernal

Word Name:	LIST
Type:	Standard word
Data Stack Signature:	block --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Displays block *block* to the screen.
Example:	none
Comment:	LIST uses BLOCK to load the block into memory, which may in term invoke FLUSH if buffer space is required.
See Also:	BLOCK FLUSH

LOAD

In Block I/O Words in TurboForth Kernal

Word Name:	LOAD
Type:	Standard word
Data Stack Signature:	block --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Loads (compiles/executes) block block.
Example:	none
Comment:	none
See Also:	None listed

MKBLK

In Block I/O Words in TurboForth Kernal

Word Name:	MKBLK
Type:	Standard word
Data Stack Signature:	size --
Return Stack Signature:	--
Availability:	V1.2
Description:	Creates a block file (used for storing source code and/or data) on the nominated disk, with enough space for `size` screens/blocks of data and sets the new file as the current blocks file.
Example:	`80 MKBLK DSK1.BLOCKS` The above creates a file called BLOCKS on DSK1 (of type D/F128) with enough allocated space for 80 screens/blocks.
Comment:	**Note**: In version 1.0 and 1.1 the word `MKDSK` is used, which has a slightly different syntax (though the format of the files created is compatible). Please see `MKDSK` for V1.0 and V1.1. **Note**: After creating a block file with `MKBLK` or `MKDSK` the *newly created* block file becomes the *currently active* blocks file.
See Also:	MKDSK USE

MKDSK

In Block I/O Words in TurboForth Kernal

Word Name:	MKDSK
Type:	Immediate word
Data Stack Signature:	size --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Creates a block file with the name file and capacity for *size* blocks.
Example:	`MKDSK 80 DSK1.BLOCKS` The above creates a blocks file (of type D/F128) on DSK1 called BLOCKS with enough space for 80 screens/blocks.
Comment:	**Note**: In version 1.0 & 1.1 this command was called MKDSK. **Note**: The syntax of this word has been changed for version 1.2. Please see MKBLK for V1.2. **Note**: After creating a block file with `MKBLK` or `MKDSK` the *newly created* block file becomes the *currently active* blocks file.
See Also:	USE MKBLK

SETBLK

In Block I/O Words in TurboForth Kernal

Word Name:	SETBLK
Type:	Standard word
Data Stack Signature:	buffer block --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Associates buffer *buffer* with block *block*.
Example:	none
Comment:	none
See Also:	BLK? BUF? CLEAN DIRTY DIRTY? SETBLK

THRU

In Block I/O Words in TurboForth Kernal

Word Name:	THRU
Type:	Standard word
Data Stack Signature:	start end --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Loads blocks starting at block *start* thru *end* inclusive.
Example:	none
Comment:	none
See Also:	LOAD

UPDATE

In Block I/O Words in TurboForth Kernal

Word Name:	UPDATE
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Marks the last accessed block as dirty.
Example:	none
Comment:	Marks the last accessed block as dirty, such that it will be flushed to disk when FLUSH is called. FLUSH may be called either explicitly in user code, or implicitly by the kernal when all block buffers are exhausted.
See Also:	FLUSH

USE

In Block I/O Words in TurboForth Kernal

Word Name:	USE
Type:	Standard word
Data Stack Signature:	string --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Defines the block file to be used. A string should be set up on the stack before calling `USE`.
Example:	`S" DSK1.BLOCKS" USE`
Comment:	When `USE` is executed, all buffers are cleared (via `EMPTY-BUFFERS`). If any buffers are marked as changed/dirty they are *not* flushed to disk. Therefore, it is the users responsibility to `FLUSH` any buffers back to disk before executing `USE`.
See Also:	EMPTY-BUFFERS FLUSH

WHERE

In Block I/O Words in TurboForth Kernal

Word Name:	WHERE
Type:	Standard word
Data Stack Signature:	-- block
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns the block number of a user-loaded word in the dictionary. Non-existant words return 0.
Example:	`WHERE CPYBLK LIST`
Comment:	Useful in conjunction with `EDIT` and `LIST`. Note that a word must be loaded (in the dictionary) from a block before its block location can be located with `WHERE`. **Note**: In versions 1.0 and 1.1 `WHERE` was an immediate word. In version 1.2 it is not an immediate word. **Note**: System-provided words (words in the TurboForth ROM) return a value of 1. It is not useful to use `WHERE` in conjunction with a system-provided word.
See Also:	EDIT LIST

TurboForth

Category 2

Words in category Built-in Constants in glossary TurboForth Kernal

The following words are listed in this category (10 words found):

Word Types: S = Standard, I = Immediate , IC = Imme

Word Name	Word Type	Data Stack Signature	Return Stack Signature	Description	Availability V1.0	V1.1	V1.2
FALSE	S	-- 0	--	Returns 0.	☑	☑	☑
FFAIHM	S	-- n	--	Returns the first free address in high memory.	☑	☑	☑
FFAILM	S	-- n	--	Returns the first free address in low memory.	☑	☑	☑
HERE	S	-- address	--	Returns the address of the start of free memory.	☑	☑	☑
IOERR	S	-- io_error	--	Returns the last disk IO error. If 0, no error occurred.	☑	☑	☑
PAD	S	-- address	--	Returns an address for use as a temporary pad or work area.	☑	☑	☑
R0	S	-- address	--	Returns beginning address of return stack. Used to reset return stack.	☑	☑	☑
S0	S	-- address	--	Returns beginning address of data stack. Used to reset data stack.	☑	☑	☑
TRUE	S	-- -1	--	Returns -1 (>FFFF hex).	☑	☑	☑
XMAX	S	-- width	--	Returns the number of columns on the display. Either 32, 40 or 80.	☑	☑	☑

FALSE

In Built-in Constants in TurboForth Kernal

Word Name:	FALSE
Type:	Standard word
Data Stack Signature:	-- 0
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns 0.

Example:	`: IsEven? ( n -- )`
	`  \ return TRUE if word is even, else return FALSE`
	`  1 AND IF FALSE ELSE TRUE THEN`
	`;`
	
	`2 isEven? . -1 ok:0`
	`3 isEven? . 0 ok:0`

Comment:	May be used in `IF` statements etc.
See Also:	TRUE

FFAIHM

In Built-in Constants in TurboForth Kernal

Word Name:	FFAIHM
Type:	Standard word
Data Stack Signature:	-- n
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns the first free address in high memory.
Example:	none
Comment:	When compiling to high memory (i.e. when `H` is set to an address in the 24K memory expansion) `FFAIHM` tracks the next free memory address.
See Also:	H FFAILM HERE

FFAILM

In Built-in Constants in TurboForth Kernal

Word Name:	FFAILM
Type:	Standard word
Data Stack Signature:	-- n
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns the first free address in low memory.
Example:	none
Comment:	When compiling to low memory (i.e. when `H` is set to an address in the 8K memory expansion) `FFAILM` tracks the next free memory address.
See Also:	H FFAIHM HERE

HERE

In Built-in Constants in TurboForth Kernal

Word Name:	HERE
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns the address of the start of free memory.
Example:	none
Comment:	Free memory is defined as that memory that has not had code compiled into it, or data compiled into it (with ,) . HERE tracks either low memory (the 8K memory expansion) or high memory (the 24K memory expansion) depending on the address set via the variable H.
See Also:	, (comma) H FFAIHM FFAILM

IOERR

In Built-in Constants in TurboForth Kernal

Word Name:	IOERR
Type:	Standard word
Data Stack Signature:	-- io_error
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns the last disk IO error. If 0, no error occurred.
Example:	none
Comment:	none
See Also:	None listed

PAD

In Built-in Constants in TurboForth Kernal

Word Name:	PAD
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns an address for use as a temporary pad or work area.
Example:	none
Comment:	The address may be in high or low memory depending on the value of HERE when PAD is executed. PAD is calculated as HERE+80 bytes. Care should be taken when HERE points to an address close to the end of either upper or lower memory expansion, as PAD could conceviably return an address that lies outside of legal memory ranges.
See Also:	H FFAIHM FFAILM HERE

R0

In Built-in Constants in TurboForth Kernal

Word Name:	R0
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns beginning address of return stack. Used to reset return stack.
Example:	none
Comment:	Pre-decrement is used on all return stack pushes. Thus, if the return stack is set to (for example) $B000, then the first value pushed onto the stack shall be pushed to address >AFFE.
See Also:	>R R> R@

S0

In Built-in Constants in TurboForth Kernal

Word Name:	S0
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns beginning address of data stack. Used to reset data stack.
Example:	none
Comment:	Pre-decrement is used on all data stack pushes. Thus, if the dats stack is set to (for example) $B000, then the first value pushed onto the stack shall be pushed to address >AFFE.
See Also:	SP! SP@

TRUE

In Built-in Constants in TurboForth Kernal

Word Name:	TRUE
Type:	Standard word
Data Stack Signature:	-- -1
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns -1 (>FFFF hex).

Example:	`: IsEven? ( n -- )`
	`  \ return TRUE if word is even, else return FALSE`
	`  1 AND IF FALSE ELSE TRUE THEN`
	`;`
	
	`2 isEven? . -1 ok:0`
	`3 isEven? . 0 ok:0`

Comment:	May be used in IF statements etc.
See Also:	FALSE

XMAX

In Built-in Constants in TurboForth Kernal

Word Name:	XMAX
Type:	Standard word
Data Stack Signature:	-- width
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns the number of columns on the display. Either 32, 40 or 80.
Example:	`1 GMODE` `XMAX . 32 ok:0`
Comment:	none
See Also:	GMODE

TurboForth

Category 3

Words in category Built-in Variables in glossary TurboForth Kernal

The following words are listed in this category (20 words found): **Word Types:** S = Standard, I = Immediate , IC = Immediate-compiling.

Word Name	Word Type	Data Stack Signature	Return Stack Signature	Description	Availability V1.0	V1.1	V1.2
#BUF	S	addr --	--	Sets the number of block buffers reserved in VDP memory. The minimum value is 1, the maximum (and default) value is 6. `#BUF` is a variable - it returns an address. It is read with `@` and written with `!`.	☐	☐	☑
#TIB	S	-- address	--	Pushes the address of the `#TIB` variable. Contains the size of the terminal input buffer (TIB).	☑	☑	☑
>IN	S	-- address	--	Pushes the address of the current read position variable in the TIB.	☑	☑	☑
BASE	S	-- address	--	Pushes the address of the current number base variable.	☑	☑	☑
C/L	S	-- address	--	Pushes the address of the 'characters per line' variable.	☑	☑	☐
CSEN	S	-- address	--	Pushes the address of the case sensitivity variable.	☐	☐	☑
H	S	-- address	--	Pushes the address of the current compilation address pointer.	☑	☑	☑
KDEL	S	-- address	--	Pushes the address of the KDEL variable.	☐	☐	☑
KMODE	S	-- address	--	Pushes the address of the keyboard scan mode variable.	☑	☑	☑
LATEST	S	-- address	--	Pushes the address of the latest dictionary entry variable.	☑	☑	☑
SP!	S	-- address	--	Sets the address of the top of the data stack.	☑	☑	☑
SP@	S	-- address	--	Pushes the data stack's current address (the address that the top of the data stack is currently pointing to).	☑	☑	☑
SPAN	S	-- address	--	Pushes the address of the `SPAN` variable.	☑	☑	☑
SSCROLL	S	-- address	--	Pushes the address of the screen scrolling variable.	☑	☑	☑
STATE	S	-- address	--	Pushes the address of the `STATE` variable.	☑	☑	☑
TIB	S	-- address	--	Pushes the address of the terminal input buffer (TIB) pointer variable.	☑	☑	☑
UNSIGNED	S	-- addr	--	Used to determine if numbers shall be converted to signed or unsigned strings by the word `N>S`.	☐	☐	☑
WARN	S	-- address	--	Pushes the address of the WARN variable.	☑	☑	☑
WRAP	S	-- address	--	Pushes the address of the `WRAP` variable.	☑	☑	☑
ZEROS	S	-- address	--	Pushes the address of the ZEROS variable.	☑	☑	☑

#BUF

In Built-in Variables in TurboForth Kernal

Word Name:	#BUF
Type:	Standard word
Data Stack Signature:	addr --
Return Stack Signature:	--
Availability:	V1.2
Description:	Sets the number of block buffers reserved in VDP memory. The minimum value is 1, the maximum (and default) value is 6. `#BUF` is a variable - it returns an address. It is read with `@` and written with `!`.
Example:	None
Comment:	The table below shows the effect on VDP memory for the 6 available values for #BUF. See the VDP memory map for more information.

#BUF Values

#BUF Value	Description
1	Reserves a single 1024 byte block buffer at VDP address >3020.
2	Reserves 2 block buffers in VDP memory from >2C20 to >341F.
3	Reserves 3 block buffers in VDP memory from >2820 to >341F.
4	Reserves 4 block buffers in VDP memory from >2420 to >341F.
5	Reserves 5 block buffers in VDP memory from >2020 to >341F.
6	Reserves 6 block buffers in VDP memory from >1C20 to >341F.

See Also:	None listed

#TIB

In Built-in Variables in TurboForth Kernal

Word Name:	#TIB
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Pushes the address of the `#TIB` variable. Contains the size of the terminal input buffer (TIB).
Example:	none
Comment:	Default size is 80 bytes.
See Also:	EXPECT SPAN TIB

>IN

In Built-in Variables in TurboForth Kernal

Word Name:	>IN
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Pushes the address of the current read position variable in the TIB.
Example:	none
Comment:	As the Terminal Input Buffer is read with WORD, the value in >IN advances to point to the beginning of the next part of the TIB to be read. Calls to EXPECT shall reset >IN to 0. When >IN is equal to the value in #TIB then all input has been exausted and WORD shall return 0 0.
See Also:	EXPECT WORD #TIB

BASE

In Built-in Variables in TurboForth Kernal

Word Name:	BASE
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Pushes the address of the current number base variable.

Example:	`: TEST`
	`  10 0 DO I . LOOP ;`
	
	`DECIMAL TEST`
	`2 BASE ! TEST`

Comment:	Default value of BASE is 10 (decimal). Setting BASE changes how numbers are interpreted in text entered via the command line, and also in source code. The allowable range for BASE is 2 to 36. Any value outside of this range shall cause unpredictable results.
See Also:	DECIMAL HEX

C/L

In Built-in Variables in TurboForth Kernal

Word Name:	C/L
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1
Description:	Pushes the address of the 'characters per line' variable.
Example:	none
Comment:	C/L is 80 when reading from the command line or 64 when reading a block.
See Also:	#TIB SPAN

CSEN

In Built-in Variables in TurboForth Kernal

Word Name:	CSEN
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.2
Description:	Pushes the address of the case sensitivity variable.
Example:	`-1 CSEN ! \ case sensitive` `0 CSEN ! \ non case sensitive`
Comment:	When CSEN is -1 (or TRUE) all dictionary searches shall be case sensitive. Default value is 0 (FALSE) - not case sensitive.
See Also:	None listed

H

In Built-in Variables in TurboForth Kernal

Word Name:	H
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Pushes the address of the current compilation address pointer.
Example:	none
Comment:	H is used to track the next free memory address to be compiled to. The compilation address can be changed by writing to this variable. All words which cause something to be *compiled* to memory (e.g. `:` , `CREATE` `VARIABLE` etc) cause `H` to be changed (advanced).
See Also:	, (comma) : (colon) ALIGN ALLOT C, COMPILE CONSTANT CREATE VARIABLE FFAIHM FFAILM HERE

KDEL

Search: Search

In Built-in Variables in TurboForth Kernal

Word Name:	KDEL
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.2
Description:	Pushes the address of the KDEL variable.
Example:	None
Comment:	KDEL is used for setting (via !) and reading (via @) the keyboard auto-repeat rate to be used in the block editor. This is intended for Geneve compatibility as the editor is unusable on the Geneve due to the keyboard delays being too short. This work-around poses a neat fix. Geneve users can set the delay by writing to KDEL in their BLOCKS startup file, or by typing directly at the command line (the command line works fine on the Geneve. The value stored in KDEL has the following format: `0xIISS` Where II is the initial delay (before auto-repeat starts) divided by 2. At power up, this value is 0xED. SS is the short delay (when auto-repeat is active). At power-up this value is 0x1E. Maximum value is FF. **Note:** Introduced in TurboForth V1.2.1:4 (September 2015)
See Also:	None listed

KMODE

In Built-in Variables in TurboForth Kernal

Word Name:	KMODE
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Pushes the address of the keyboard scan mode variable.
Example:	$0500 KMODE ! \ sets the keyboard mode. KMODE @ \ fetches the keyboard mode to the stack.
Comment:	Setting this mode influences the ASCII codes returned by the TI KSCAN ROM routine. Default value is 5. Valid values are: • $0100 - scans the left side of the keyboard • $0200 - scans the right side of the keyboard • $0300 - TI-99/4 scan mode. In this mode, lower case characters are returned as upper case characters. FCTN codes 1 thru 15 are returned. Control characters are ignored. • $0400 - Pascal mode. Upper and lower case characters are returned. FCTN keys return codes in the range 129 thru 143. Control characters return codes in the range 1 thru 31. • $0500 - BASIC mode (TurboForth default). Upper and lower case characters are returned. FCTN keys return codes from 1 thru 15. Control keys return codes from 128 thru 159 (1nd 187). **Note:** TurboForth sets KMODE to $0500 (the default) whenever the command line is active. **Note:** The keyboard mode should be placed in the high-byte.
See Also:	JOYST KEY KEY?

LATEST

In Built-in Variables in TurboForth Kernal

Word Name:	LATEST
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Pushes the address of the latest dictionary entry variable.
Example:	none
Comment:	The value in `LATEST` points to the start (the link field) of the last entry in the dictionary. Words such as `: CODE: CREATE VARIABLE VALUE and CONSTANT` cause dictionary entries to be created, thus cause `LATEST` to be updated.
See Also:	: (colon) CODE: CONSTANT CREATE VALUE VARIABLE

SP!

In Built-in Variables in TurboForth Kernal

Word Name:	SP!
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Sets the address of the top of the data stack.
Example:	`$B000 SP!` (sets the base of the stack to B000 hex)
Comment:	none
See Also:	SP@ S0

SP@

In Built-in Variables in TurboForth Kernal

Word Name:	SP@
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Pushes the data stack's current address (the address that the top of the data stack is currently pointing to).
Example:	`100 SP@ @ .` (displays 100)
Comment:	none
See Also:	SP! S0

SPAN

In Built-in Variables in TurboForth Kernal

Word Name:	SPAN
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Pushes the address of the SPAN variable.
Example:	none
Comment:	SPAN contains the number of characters received by EXPECT.
See Also:	EXPECT WORD #TIB C/L TIB

SSCROLL

In Built-in Variables in TurboForth Kernal

Word Name:	SSCROLL
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Pushes the address of the screen scrolling variable.
Example:	none
Comment:	Determines if screen scrolling shall be used. Defaults to TRUE. When set to FALSE screen scrolling shall be disabled, and screen output shall resume at the top of the screen when the last screen line has been reached.
See Also:	None listed

STATE

In Built-in Variables in TurboForth Kernal

Word Name:	STATE
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Pushes the address of the `STATE` variable.
Example:	none
Comment:	When `STATE`=0, the compiler/interpreter is interpreting, otherwise it is compiling.
See Also:	INTERPRET

TIB

In Built-in Variables in TurboForth Kernal

Word Name:	TIB
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Pushes the address of the terminal input buffer (TIB) pointer variable.
Example:	none
Comment:	**Important note:** • In versions 1.0 and 1.1, the address that TIB returns (i.e. with the phrase TIB @) refers to starting addess of a buffer in CPU memory. • In vesion 1.2, the address that TIB returns (i.e. with the phrase TIB @) refers to starting addess of a buffer in VDP memory. • In version 1.2 EXPECT places characters into VDP memory. • In version 1.2 WORD copies a word from VDP memory into a buffer in CPU memory.
See Also:	EXPECT WORD #TIB C/L SPAN

UNSIGNED

In Built-in Variables in TurboForth Kernal

Word Name:	UNSIGNED
Type:	Standard word
Data Stack Signature:	-- addr
Return Stack Signature:	--
Availability:	V1.2
Description:	Used to determine if numbers shall be converted to signed or unsigned strings by the word N>S.

Example:	`TRUE UNSIGNED !`
	`-1 N>S TYPE CR`
	
	(displays 65535)
	
	`FALSE UNSIGNED !`
	`-1 N>S TYPE CR`
	
	(displays -1)

Comment:	When executed, returns the address of the UNSIGNED variable so that it can be accessed as a variable with ! and @. Internally, words such as . U. .R and U.R manipulate the UNSIGNED variable in order to force N>S (number to string) to convert the numeric value in the appropriate signed or unsigned format.
See Also:	ZEROS . .R N>S

WARN

In Built-in Variables in TurboForth Kernal

Word Name:	WARN
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Pushes the address of the WARN variable.
Example:	none
Comment:	Word re-definition warnings are suppressed when WARN=0 Currently, only colon-definitions issue a warning upon re-definition (when WARN>0). Any other type of defining word is not checked.
See Also:	None listed

WRAP

In Built-in Variables in TurboForth Kernal

Word Name:	WRAP
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Pushes the address of the `WRAP` variable.

Example:	`: TEST`
	`  10 10 10 10 PANEL \ define a panel`
	`  TRUE WRAP ! \ select panel wrapping`
	`  100 0 DO 2 SCROLL LOOP ( scroll screen to the right) ;`

Comment:	Used with `SCROLL` and `PANEL`. If TRUE, `SCROLL` will do a wrapping scroll.
See Also:	PANEL SCROLL

ZEROS

In Built-in Variables in TurboForth Kernal

Word Name:	ZEROS
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Pushes the address of the ZEROS variable.
Example:	none
Comment:	If `ZEROS`=TRUE, `. U.` and `$.` etc will display leading zeros.
See Also:	None listed

TurboForth

Category 4

Words in category Comparison Words in glossary TurboForth Kernal

The following words are listed in this category (14 words found): **Word Types:** S = Standard, I = Immediate , IC = Immediate-compiling.

Word Name	Word Type	Data Stack Signature	Return Stack Signature	Description	Availability V1.0	V1.1	V1.2
<	S	a b -- true|false	--	Replaces a and b with TRUE if a < b, else replaces with FALSE.	☑	☑	☑
<=	S	a b -- true|false	--	Replaces a and b with TRUE if a < b, else replaces with FALSE.	☑	☑	☑
<>	S	a b -- true|false	--	Replaces a and b with TRUE if a and b are of different value, else replaces with FALSE.	☑	☑	☑
=	S	a b -- true|false	--	Replaces a and b with TRUE if a and b are equal, else replaces with FALSE.	☑	☑	☑
>	S	a b -- true|false	--	Replaces a and b with TRUE if a > b, else replaces with FALSE.	☑	☑	☑
>=	S	a b -- true|false	--	Replaces a and b with TRUE if a >= b, else replaces with FALSE.	☑	☑	☑
0<	S	a -- true|false	--	Replaces a with TRUE if a<0, else replaces with FALSE.	☑	☑	☑
0<=	S	a -- true|false	--	Replaces a with TRUE is a<=0, else replaces with FALSE.	☑	☑	☑
0<>	S	a -- true|false	--	Replaces a with TRUE is a<>0, else replaces with FALSE.	☑	☑	☑
0=	S	a -- true|false	--	Replaces a with TRUE if a=0, else replaces with FALSE.	☑	☑	☑
0>	S	a -- true|false	--	Replaces a with TRUE if a>0, else replaces with FALSE.	☑	☑	☑
0>=	S	a -- true|false	--	Replaces a with TRUE if a>=0, else replaces with FALSE.	☑	☑	☑
U<	S	a b -- true|false	--	Performs an unsigned < comparison.	☑	☑	☑
WITHIN	S	n low high -- true|false	--	Replaces n, low, and high with TRUE if n is >= low and < high, else replaces with FALSE.	☑	☑	☑

<

In Comparison Words in TurboForth Kernal

Word Name:	<
Type:	Standard word
Data Stack Signature:	a b -- true|false
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Replaces a and b with TRUE if a < b, else replaces with FALSE.
Example:	none
Comment:	none
See Also:	<=

<=

In Comparison Words in TurboForth Kernal

Word Name:	<=
Type:	Standard word
Data Stack Signature:	a b -- true|false
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Replaces a and b with TRUE if a < b, else replaces with FALSE.
Example:	none
Comment:	none
See Also:	≤

<>

In Comparison Words in TurboForth Kernal

Word Name:	<>
Type:	Standard word
Data Stack Signature:	a b -- true|false
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Replaces a and b with TRUE if a and b are of different value, else replaces with FALSE.
Example:	none
Comment:	none
See Also:	≤ <= ≥ >=

=

In Comparison Words in TurboForth Kernal

Word Name:	=
Type:	Standard word
Data Stack Signature:	a b -- true|false
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Replaces a and b with TRUE if a and b are equal, else replaces with FALSE.
Example:	none
Comment:	none
See Also:	<>

>

In Comparison Words in TurboForth Kernal

Word Name:	>
Type:	Standard word
Data Stack Signature:	a b -- true|false
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Replaces a and b with TRUE if a > b, else replaces with FALSE.
Example:	none
Comment:	none
See Also:	>=

>=

In Comparison Words in TurboForth Kernal

Word Name:	>=
Type:	Standard word
Data Stack Signature:	a b -- true|false
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Replaces a and b with TRUE if a >= b, else replaces with FALSE.
Example:	none
Comment:	none
See Also:	>

0<

In Comparison Words in TurboForth Kernal

Word Name:	0<
Type:	Standard word
Data Stack Signature:	a -- true|false
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Replaces a with TRUE if a<0, else replaces with FALSE.
Example:	none
Comment:	none
See Also:	0<= 0<> 0= 0> 0>= 0!

0<=

In Comparison Words in TurboForth Kernal

Word Name:	0<=
Type:	Standard word
Data Stack Signature:	a -- true|false
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Replaces a with TRUE is a<=0, else replaces with FALSE.
Example:	none
Comment:	none
See Also:	0< 0<> 0= 0> 0>=

0<>

In Comparison Words in TurboForth Kernal

Word Name:	0<>
Type:	Standard word
Data Stack Signature:	a -- true|false
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Replaces a with TRUE is a<>0, else replaces with FALSE.
Example:	none
Comment:	none
See Also:	0< 0<= 0= 0> 0>=

0=

In Comparison Words in TurboForth Kernal

Word Name:	0=
Type:	Standard word
Data Stack Signature:	a -- true|false
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Replaces a with TRUE if a=0, else replaces with FALSE.
Example:	none
Comment:	none
See Also:	0< 0<= 0<> 0> 0>=

0>

In Comparison Words in TurboForth Kernal

Word Name:	0>
Type:	Standard word
Data Stack Signature:	a -- true|false
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Replaces a with TRUE if a>0, else replaces with FALSE.
Example:	none
Comment:	none
See Also:	0< 0<= 0<> 0= 0>=

0>=

In Comparison Words in TurboForth Kernal

Word Name:	0>=
Type:	Standard word
Data Stack Signature:	a -- true|false
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Replaces a with TRUE if a>=0, else replaces with FALSE.
Example:	none
Comment:	none
See Also:	0< 0<= 0<> 0= 0>

U<

In Comparison Words in TurboForth Kernal

Word Name:	U<
Type:	Standard word
Data Stack Signature:	a b -- true|false
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Performs an unsigned < comparison.
Example:	none
Comment:	Replaces *a* and *b* with TRUE if unsigned value *a* is less than unsigned value *b*, else replaces with FALSE. Note: No version of U> is present, however U< can be used to derive U> as follows: `: U> ( a b -- flag) SWAP U< ;`
See Also:	<

WITHIN

In Comparison Words in TurboForth Kernal

Word Name:	WITHIN
Type:	Standard word
Data Stack Signature:	n low high -- true|false
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Replaces n, low, and high with TRUE if n is >= low and < high, else replaces with FALSE.
Example:	`99 40 400 WITHIN .` \ returns true, since 99 is within the range of 40 to 400 `99 10 20 WITHIN .` \ returns false `-20 -100 -1 WITHIN .` \ returns true
Comment:	*none*
See Also:	MAX MIN

TurboForth

Category 5

Words in category Compilation Words in glossary TurboForth Kernal

The following words are listed in this category (29 words found): **Word Types:** S = Standard, I = Immediate , IC = Immediate-compiling.

Word Name	Word Type	Data Stack Signature	Return Stack Signature	Description	Availability V1.0	V1.1	V1.2
' (tick)	S	-- xt	--	Returns the XT/CFA of the word immediately following.	☑	☑	☑
, (comma)	S	n --	--	Compiles the 16 bit cell n on the stack to HERE.	☑	☑	☑
: (colon)	SC	--	--	Calls HEADER to build a dictionary entry and begins compilation of a colon definition (sets STATE to 1).	☑	☑	☑
; (semi-colon)	IC	--	--	Ends compilation of the current colon definition.	☑	☑	☑
;CODE	IC	--	--	Terminates the definition of a machine code word.	☑	☑	☑
[	I	--	--	Temporarily suspends compilation (sets STATE to 0) and begins interpreting input as immediate mode input.	☑	☑	☑
[']	I	-- xt	--	Compiles the word following ['] as a literal.	☑	☑	☑
[COMPILE]	IC	--	--	Compiles the XT/CFA of the word immediately following it to the current colon definition at address HERE.	☑	☑	☑
]	I	--	--	Ends interpretation mode and resumes normal compilation. See [.	☑	☑	☑
+TO	IC	value --	--	Adds value to the value in the VALUE immediately following it.	☑	☑	☑
ALIGN	SC	--	--	Aligns H to the next (highest) even compilation address. Takes no action if H is already even.	☑	☑	☑
ALLOT	SC	n --	--	Allots n bytes by advancing HERE by n bytes.	☑	☑	☑
C,	SC	n --	--	Compiles the lower 8 bits of the cell n on the stack to HERE. Advances HERE by 1 byte.	☑	☑	☑
CODE:	SC	--	--	Begins compilation of a machine code/assembly language word.	☑	☑	☑
COMPILE	SC	--	--	Compiles the XT/CFA of the word immediately following it to the current colon definition at address HERE.	☑	☑	☑
CONSTANT	I	n --	--	Creates a CONSTANT with the value *n*.	☑	☑	☑
CREATE	SC	--	--	Reads forward in the terminal input buffer and creates a word in the dictionary whose run-time behaviour is to return the body address of the newly created word.	☑	☑	☑
DOES>	IC	--	--	Links a run-time action to a word created with CREATE.	☑	☑	☑
EXECUTE	S	xt --	--	Executes the word whose XT/CFA is on the stack.	☑	☑	☑
HEADER	S	address length --	--	Creates a dictionary entry with the name of the string nominated at address and length.	☐	☐	☑
HEADER (V1.2.1)	S	--	--	This description pertains to HEADER in TurboForth **Version 1.2.1 only**. HEADER searches forward in the input buffer for the next available word and creates and links a new dictionary entry accordingly.	☐	☐	☑
HIDDEN	S	da --	--	Toggles the hidden bit on/off in the last created dictionary entry pointed to by dictionary address da.	☑	☑	☑
IMMEDIATE	S	--	--	Toggles the immediate bit on/off in the last created word.	☑	☑	☑
LIT	S	-- n	--	At execution time, places the value in the cell immediately following the LIT instruction onto the data stack.	☑	☑	☑
LITERAL	IC	n --	--	Compiles LIT n to the current definition. n is a 16-bit cell.	☑	☑	☑
RECURSE	IC	--	--	Causes a recursive jump to the beginning of the word being defined.	☑	☑	☑
TO	IC	n --	--	Used with a VALUE. Replaces the value of the named VALUE with n.	☑	☑	☑
VALUE	IC	n --	--	Defines a VALUE using the name immediately following in the terminal input buffer with the value n.	☑	☑	☑
VARIABLE	IC	--	--	Creates a variable using the name immediately following in the terminal input buffer.	☑	☑	☑

' (tick)

In Compilation Words in TurboForth Kernal

Word Name:	' (tick)
Type:	Standard word
Data Stack Signature:	-- xt
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns the XT/CFA of the word immediately following.
Example:	none
Comment:	Use on command line, or within a [...] sequence only.
See Also:	[']

, (comma)

In Compilation Words in TurboForth Kernal

Word Name:	, (comma)
Type:	Standard word
Data Stack Signature:	n --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Compiles the 16 bit cell n on the stack to HERE.
Example:	none
Comment:	Advances HERE by 1 cell (2 bytes).
See Also:	DATA HERE

: (colon)

In Compilation Words in TurboForth Kernal

Word Name:	: (colon)
Type:	Standard compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Calls HEADER to build a dictionary entry and begins compilation of a colon definition (sets STATE to 1).
Example:	`: TEST 1 2 3 . . . ;`
Comment:	: is used to begin a word, or, to borrow a phrase from other languages, a subroutine. : must be followed by a space, then the name of the word/subroutine. Normal Forth code, and numbers (literal) may then follow. The definition ends with ; (semi-colon).
See Also:	; (semi-colon)

; (semi-colon)

In Compilation Words in TurboForth Kernal

Word Name:	; (semi-colon)
Type:	Immediate compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Ends compilation of the current colon definition.
Example:	`: TEST 1 2 3 . . . ;` `: TEST 1 2 3 DO I . ; \ causes error`
Comment:	Ends compilation of the current colon definition and un-hides the definition from the dictionary, sets `STATE` to 0. Checks `DO/LOOP FOR/NEXT` and `BEGIN/REPEAT` etc. reference counters and errors if an un-balanced construct is detected.
See Also:	: (colon)

;CODE

In Compilation Words in TurboForth Kernal

Word Name:	;CODE
Type:	Immediate compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Terminates the definition of a machine code word.
Example:	none
Comment:	Appends a B *R12 instruction to the word being defined. See `CODE:`
See Also:	CODE:

[

In Compilation Words in TurboForth Kernal

Word Name:	[
Type:	Immediate word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Temporarily suspends compilation (sets STATE to 0) and begins interpreting input as immediate mode input.
Example:	`: TEST 1 2 3 [ S" This string will not be compiled" TYPE ] . . . ;` `TEST`
Comment:	none
See Also:	]

[']

In Compilation Words in TurboForth Kernal

Word Name:	[']
Type:	Immediate word
Data Stack Signature:	-- xt
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Compiles the word following `[']` as a literal.
Example:	`: TEST ['] DUP ;` The above example will leave the execution token of DUP on the stack when TEST is executed.
Comment:	During compilation of a colon definition, compiles the word following `[']` as a literal. When the colon definition is executed, the word's execution token (XT/CFA) shall be pushed to the stack, and not executed.
See Also:	' (tick)

[COMPILE]

In Compilation Words in TurboForth Kernal

Word Name:	[COMPILE]
Type:	Immediate compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Compiles the XT/CFA of the word immediately following it to the current colon definition at address `HERE`.
Example:	`: ENDIF ( -- ) [COMPILE] THEN ; IMMEDIATE`
Comment:	Used to compile immediate words into the current definition.
See Also:	, (comma) COMPILE

]

In Compilation Words in TurboForth Kernal

Word Name:	]
Type:	Immediate word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Ends interpretation mode and resumes normal compilation. See `[`.
Example:	none
Comment:	none
See Also:	[

+TO

In Compilation Words in TurboForth Kernal

Word Name:	+TO
Type:	Immediate compiling word
Data Stack Signature:	value --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Adds value to the value in the VALUE immediately following it.

Example:	`VALUE SCORE`
	`98 TO SCORE`
	`2 +TO SCORE`
	`SCORE .`
	
	(displays 100)

Comment:	+TO does not verify that the VALUE following it actually exists. Unpredictable results may occur if a non existant word is supplied.
See Also:	TO VALUE

ALIGN

In Compilation Words in TurboForth Kernal

Word Name:	ALIGN
Type:	Standard compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Aligns H to the next (highest) even compilation address. Takes no action if H is already even.

Example:	`ALIGN`
	`99 C,`
	`H @ $.`
	
	(TurboForth displays an odd value)
	
	`ALIGN H @ $.`
	
	(TurboForth aligns H/HERE, displaying an even value)

Comment:	Dictionary entries and executable code must align on even addresses. HEADER automatically calls ALIGN before creating dictionary entries, but sometimes H or HERE can be left with an odd value. The most common cause of this is the word C, which compiles a single byte to memory. In such situations where an odd value in H/HERE is undesirable, ALIGN will align H to the next even value. Note that if H is already aligned to an even address then ALIGN shall take no action.
See Also:	,(comma) C, HEADER H HERE

ALLOT

In Compilation Words in TurboForth Kernal

Word Name:	ALLOT
Type:	Standard compiling word
Data Stack Signature:	n --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Allots n bytes by advancing HERE by n bytes.
Example:	none
Comment:	Note: If an odd number of bytes is requested, H/HERE will be aligned to the next highest even address.
See Also:	ALIGN H HERE

C,

In Compilation Words in TurboForth Kernal

Word Name:	C,
Type:	Standard compiling word
Data Stack Signature:	n --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Compiles the lower 8 bits of the cell n on the stack to HERE. Advances HERE by 1 byte.
Example:	none
Comment:	none
See Also:	, (comma) ALIGN H HERE

CODE:

In Compilation Words in TurboForth Kernal

Word Name:	CODE:
Type:	Standard compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Begins compilation of a machine code/assembly language word.

Example:	Example (for TurboForth V1.2):
	`HEX CODE: PLUS A534 ;CODE DECIMAL`
	Creates a machine code word called PLUS which contains the following machine code:
	`A *R4+,*R4 ; pop top of stack and add to next number on stack`
	`B *R12 ; call NEXT`
	To test it, try
	`9 100 PLUS .`

Comment:	The name of the word must follow `CODE:` followed by a sequence of numbers representing the machine code op-codes. The numbers will be interpreted to according the current value of `BASE`. `CODE:` writes the dictionary entry in a format suitable for machine code words (the Code Field Address (CFA) points to the machine code, instead of DOCOL) and then compiles the op-codes into consecutive addresses in memory with no translation or conversions applied. The machine code should be terminated with `;CODE`. Upon termination, `;CODE` appends a B *R12 instruction to the compiled code, which (at run time) calls NEXT.
See Also:	;CODE

COMPILE

In Compilation Words in TurboForth Kernal

Word Name:	COMPILE
Type:	Standard compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Compiles the XT/CFA of the word immediately following it to the current colon definition at address `HERE`.
Example:	none
Comment:	To force the compilation of an immediate word use `[COMPILE]`.
See Also:	' (tick) ['] [COMPILE]

CONSTANT

In Compilation Words in TurboForth Kernal

Word Name:	CONSTANT
Type:	Immediate word
Data Stack Signature:	n --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Creates a `CONSTANT` with the value *n*.

Example:	`10 CONSTANT TEN`
	`TEN .`
	
	(displays 10)

Comment:	The name is taken from the terminal input buffer. **CONSTANTs should not be defined within a colon definition.**
See Also:	VALUE VARIABLE

CREATE

In Compilation Words in TurboForth Kernal

Word Name:	CREATE
Type:	Standard compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Reads forward in the terminal input buffer and creates a word in the dictionary whose run-time behaviour is to return the body address of the newly created word.

Example:	`CREATE FRED 99 ,`
	`FRED @ .`
	
	(displays 99)

Comment:	none
See Also:	>BODY DOES>

DOES>

In Compilation Words in TurboForth Kernal

Word Name:	DOES>
Type:	Immediate compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Links a run-time action to a word created with CREATE.

Example:	`: MALE ( age -- ) CREATE , DOES> ." Is male and is " @ . ." years old." CR ;`
	`: FEMALE ( age -- ) CREATE , DOES> ." Is female and is " @ . ." years old." CR ;`
	
	`42 MALE TOM`
	`27 FEMALE SALLY`
	
	`TOM`
	
	(TurboForth displays "TOM is male and is 42 years old")
	
	`SALLY`
	
	(TurboForth displays "SALLY is female and is 27 years old")
	
	Thus TOM is a member of the class *MALE*, and *SALLY* is a member of the class FEMALE.

Comment:	The run-time action of words created via CREATE is to leave the address of their Parameter Field (PFA) on the stack. The PFA is one cell wide and can be used to store any 16-bit value. DOES> can be used to assign run-time code to words created with CREATE. Multiple words, all created with CREATE can inherit the same run-time code/behaviour via DOES>, allowing simple classes of words to be created.
See Also:	CREATE

EXECUTE

In Compilation Words in TurboForth Kernal

Word Name:	EXECUTE
Type:	Standard word
Data Stack Signature:	xt --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Executes the word whose XT/CFA is on the stack.

Example:	`: TEST ( -- ) 1 2 3 . . . ;`
	`' TEST EXECUTE`
	
	(executes TEST)

Comment:	none
See Also:	' (tick) [']

HEADER

In Compilation Words in TurboForth Kernal

Word Name:	HEADER
Type:	Standard word
Data Stack Signature:	address length --
Return Stack Signature:	--
Availability:	V1.2
Description:	Creates a dictionary entry with the name of the string nominated at address and length.
Example:	`S" TEST" HEADER`
Comment:	The new dictionary entry is automatically linked to the previous dictionary entry, and `LATEST` is updated to point to the link field of the new dictionary entry. Note that no executable code is appended to the definition at this point. `HEADER` only creates a dictionary entry. `HEADER` is used by words such as `: CODE: VARIABLE VALUE CREATE` and `CONSTANT` to create dictionary headers on their behalf. **Important note:** In TurboForth V1.2.1 `HEADER` has a different stack signature and behaviour. Please see the seperate entry for V1.2.1 version.
See Also:	HEADER (V1.2.1)

HEADER (V1.2.1)

In Compilation Words in TurboForth Kernal

Word Name:	HEADER (V1.2.1)
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.2
Description:	This description pertains to HEADER in TurboForth **Version 1.2.1 only**. HEADER searches forward in the input buffer for the next available word and creates and links a new dictionary entry accordingly.
Example:	`HEADER NEW_WORD` The above example creates a new dictionary entry *only* (no executable code is associated with it at this point) called NEW_WORD. Previous versions of TurboForth have a different version of `HEADER`.
Comment:	None
See Also:	HEADER

HIDDEN

In Compilation Words in TurboForth Kernal

Word Name:	HIDDEN
Type:	Standard word
Data Stack Signature:	da --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Toggles the hidden bit on/off in the last created dictionary entry pointed to by dictionary address da.

Example:	`: TEST ( -- ) 1 2 3 ;`
	`' TEST >LINK HIDDEN`
	`TEST`
	
	`ERROR:`
	`TEST not found`
	
	Note: At this point, it is not possible to re-instate the word by means of tick (`'`) since the word is now hidden, and `'` cannot therefore find it. To re-instate it, the address would have to be saved elsewhere.

Comment:	When a word is hidden, dictionary searches with `FIND` will not find the word. Colon (`:`) hides new words during compilation so that new words may be defined in terms of identically named older words. Hiding the definition ensures `FIND` finds the old definition, not the new one. Semi-colon (`;`) un-hides the new definition (by executing `HIDDEN` again) at the end of compilation.
See Also:	>LINK ' (tick)

IMMEDIATE

In Compilation Words in TurboForth Kernal

Word Name:	IMMEDIATE
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Toggles the immediate bit on/off in the last created word.

Example:	`: BOO! ( -- ) CR ." Boo! I am immediate. I execute at compile time" CR ; IMMEDIATE`
	`: TEST ( -- ) 1 2 3 BOO! . . . ;`
	
	`Boo! I am immediate. I execute at compile time`
	
	TEST
	`3 2 1 ok:0`

Comment:	An immediate word will be executed (rather than compiled) when it is encountered during the compilation of a colon definition.
See Also:	: (colon)

LIT

In Compilation Words in TurboForth Kernal

Word Name:	LIT
Type:	Standard word
Data Stack Signature:	-- n
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	At execution time, places the value in the cell immediately following the LIT instruction onto the data stack.

Example:	`: TEST ( -- ) 1 2 3 ;`
	Causes the following sequence to be compiled into TEST:
	`LIT 1 LIT 2 LIT 3`
	At run time, `LIT` pushes the value in front of it to the data stack.

Comment:	`LIT` is compiled into definitions by the compiler when a number is encountered in a definition. Also see `LITERAL`.
See Also:	LITERAL

LITERAL

In Compilation Words in TurboForth Kernal

Word Name:	LITERAL
Type:	Immediate compiling word
Data Stack Signature:	n --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Compiles LIT n to the current definition. n is a 16-bit cell.

Example:	If A B and C are constants, then:
	`: GET-RESULT A B + C * ;`
	can be more efficiently expressed as:
	`: GET-RESULT [ A B + C * ] LITERAL ;`
	In the first example, (A+B)*C is repeatedly calculated each time GET-RESULT is called. In the second example, (A+B)*C is evaluated in immediate mode (facilitated by `[` and `]`) and encoded into the colon definition as a literal by means of `LITERAL`.

Comment:	Normally used in conjuction with `[` and `]` to encode the result of a calculation into a colon definition as a literal (to save having to evaluate the calculation at run-time).
See Also:	[ ] LIT

RECURSE

In Compilation Words in TurboForth Kernal

Word Name:	RECURSE
Type:	Immediate compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Causes a recursive jump to the beginning of the word being defined.
Example:	none
Comment:	Since new words (in the process of being created) are hidden, they can not be referenced with their own name. Thus `RECURSE` creates a recursive call to the word currently under compilation.
See Also:	None listed

TO

In Compilation Words in TurboForth Kernal

Word Name:	TO
Type:	Immediate compiling word
Data Stack Signature:	n --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Used with a VALUE. Replaces the value of the named VALUE with n.

Example:	`0 VALUE SCORE`
	`100 TO SCORE`
	`SCORE .`
	
	(displays 100)

Comment:	`TO` does not verify that the `VALUE` following it actually exists. Unpredictable results may occur if a non existant word is supplied.
See Also:	+TO VALUE

VARIABLE

In Compilation Words in TurboForth Kernal

Word Name:	VARIABLE
Type:	Immediate compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Creates a variable using the name immediately following in the terminal input buffer.
Example:	`VARIABLE DELAY` `100 DELAY !` (stores 100 in the variable DELAY) `DELAY @ .` (fetches the variables value and displays it (displays 100))
Comment:	Variables are read and set with `@` and `!`. `VARIABLE`**s should not be defined within a colon definition.**
See Also:	! @ CONSTANT VALUE

TurboForth

Category 6

Words in category Console I/O Words in glossary TurboForth Kernal

The following words are listed in this category (14 words found): **Word Types:** S = Standard, I = Immediate , IC = Immediate-compiling.

Word Name	Word Type	Data Stack Signature	Return Stack Signature	Description	Availability V1.0	V1.1	V1.2
BREAK?	S	--	--	Scans keyboard for break (FCTN & 4).	☑	☑	☑
CR	S	--	--	Issues a carriage return to the screen cursor (begins a new line).	☑	☑	☑
EMIT	S	n --	--	Emits ASCII character n to the screen to the current cursor position.	☑	☑	☑
GOTOXY	S	x y --	--	Sets screen cursor location to column *x* and row *y*.	☑	☑	☑
JOYST	S	unit# -- value	--	Scans the selected joystick.	☑	☑	☑
KEY	S	-- ascii	--	Suspends execution and waits for a key press. Returns ascii code of the key pressed.	☑	☑	☑
KEY?	S	-- ascii|-1	--	Scans keyboard and returns ascii code if a key is detected, else returns -1.	☑	☑	☑
PAGE	S	--	--	Clears the screen.	☑	☑	☑
SPACE	S	--	--	EMITs a single space character (ascii 32) to the screen.	☑	☑	☑
SPACES	S	n --	--	EMITs *count* spaces to the screen.	☑	☑	☑
TERMINAL?	S	-- flag	--	Scans keyboard and tests for break (FCTN & 4). Returns TRUE if detected, else returns FALSE.	☑	☑	☐
TYPE	S	address length --	--	Types the string of length *length* beginning at address *address* to the screen.	☑	☑	☑
WORDS	S	--	--	Displays all words currently defined in the dictionary to the screen.	☑	☑	☑
XY?	S	-- x y	--	Returns the current *x* (column) and *y* (row) position of the screen cursor.	☑	☑	☑

BREAK?

In Console I/O Words in TurboForth Kernal

Word Name:	BREAK?
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Scans keyboard for break (FCTN & 4).
Example:	none
Comment:	If detected, executes `QUIT` to return to the interpreter. Any open disk files (not including blocks files) will remain open.
See Also:	KEY KEY?

CR

In Console I/O Words in TurboForth Kernal

Word Name:	CR
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Issues a carriage return to the screen cursor (begins a new line).
Example:	none
Comment:	Scrolls screen up if on the last line of the screen (the 24th row).
See Also:	GOTOXY

EMIT

In Console I/O Words in TurboForth Kernal

Word Name:	EMIT
Type:	Standard word
Data Stack Signature:	n --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Emits ASCII character n to the screen to the current cursor position.
Example:	`42 EMIT` `ASCII % EMIT`
Comment:	Advances the cursor position after executing. If the last character on the last screen row is used, the screen will be scrolled up one line (assuming the value of `SSCROLL`>0).
See Also:	TYPE ."

GOTOXY

In Console I/O Words in TurboForth Kernal

Word Name:	GOTOXY
Type:	Standard word
Data Stack Signature:	x y --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Sets screen cursor location to column *x* and row *y*.
Example:	none
Comment:	The next character to be EMITted shall be located at row *x* and column *y*.
See Also:	EMIT XY?

JOYST

In Console I/O Words in TurboForth Kernal

Word Name:	JOYST
Type:	Standard word
Data Stack Signature:	unit# -- value
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Scans the selected joystick.
Example:	none
Comment:	For joystick #1 use a unit# of **0**. For joystick 2 use a unit# of **1**. The returned value value is a bit code which can be decoded as follows: • 1=Fire • 2=Left • 4=Right • 8=Down • 16=Up Since each direction has its own bit, combinations are possible: for example, UP+LEFT+FIRE returns a value of 19. Note: The version of JOYST shipped in version 1.0 returns different values for up, down, left, right & fire. **Note:** The version of JOYST shipped in version 1.1 has a bug which can return un-predictable results or cause TurboForth to crash in some circumstances. An alternative version of JOYST can be installed in a block and used when required, as presented below: `HEX CODE: JOYST C054 0221 0006 06C1 020C 0024 30C1 020C 0006 3541 06C1 0541 0241 001F C501 020C 8328 C80C 83D6 0300 0002 0300 0000 ;CODE DECIMAL`
See Also:	KEY KEY?

KEY

In Console I/O Words in TurboForth Kernal

Word Name:	KEY
Type:	Standard word
Data Stack Signature:	-- ascii
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Suspends execution and waits for a key press. Returns ascii code of the key pressed.
Example:	none
Comment:	none
See Also:	None listed

KEY?

In Console I/O Words in TurboForth Kernal

Word Name:	KEY?
Type:	Standard word
Data Stack Signature:	-- ascii|-1
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Scans keyboard and returns ascii code if a key is detected, else returns -1.
Example:	none
Comment:	Does not suspend execution.
See Also:	JOYST KEY

PAGE

In Console I/O Words in TurboForth Kernal

Word Name:	PAGE
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Clears the screen.
Example:	none
Comment:	none
See Also:	None listed

SPACE

In Console I/O Words in TurboForth Kernal

Word Name:	SPACE
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	EMITs a single space character (ascii 32) to the screen.
Example:	none
Comment:	none
See Also:	EMIT SPACES

SPACES

In Console I/O Words in TurboForth Kernal

Word Name:	SPACES
Type:	Standard word
Data Stack Signature:	n --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	EMITs *count* spaces to the screen.
Example:	none
Comment:	none
See Also:	EMIT SPACE

TERMINAL?

In Console I/O Words in TurboForth Kernal

Word Name:	TERMINAL?
Type:	Standard word
Data Stack Signature:	-- flag
Return Stack Signature:	--
Availability:	V1.0 V1.1
Description:	Scans keyboard and tests for break (FCTN & 4). Returns TRUE if detected, else returns FALSE.
Example:	none
Comment:	TERMINAL? has been removed from version 1.2 to make space in the EPROM for other functionality. It can be defined thus: `: TERMINAL ( -- flag ) KEY? 2 = ;`
See Also:	BREAK? KEY KEY?

TYPE

In Console I/O Words in TurboForth Kernal

Word Name:	TYPE
Type:	Standard word
Data Stack Signature:	address length --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Types the string of length *length* beginning at address *address* to the screen.
Example:	`: TEST S" Hello Mother!" TYPE ;`
Comment:	none
See Also:	EMIT ." S"

WORDS

In Console I/O Words in TurboForth Kernal

Word Name:	WORDS
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Displays all words currently defined in the dictionary to the screen.
Example:	none
Comment:	Any key pauses the list. Pressing break (FCTN 4) stops the listing.
See Also:	None listed

XY?

In Console I/O Words in TurboForth Kernal

Word Name:	XY?
Type:	Standard word
Data Stack Signature:	-- x y
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns the current *x* (column) and *y* (row) position of the screen cursor.
Example:	none
Comment:	none
See Also:	GOTOXY

TurboForth

Category 7

Words in category File I/O Words in glossary TurboForth Kernal

The following words are listed in this category (10 words found): **Word Types:** S = Standard, I = Immediate , IC = Immediate-compiling.

Word Name	Word Type	Data Stack Signature	Return Stack Signature	Description	Availability V1.0	V1.1	V1.2
#CLOSE	S	file_id --	--	Closes the file with the file id file_id.	☑	☑	☑
#EOF?	S	file_id -- flag	--	Returns TRUE if file file_id is currently positioned at the end of the file.	☑	☑	☑
#GET	S	buf_addr file_id -- flag	--	Reads a record from an external file into a buffer.	☑	☑	☑
#OPEN	S	file_id -- flag	--	Opens a file with the file name and attributes specified in the buffer starting at file_id.	☑	☑	☑
#PUT	S	buf_addr len file_id -- flag	--	Writes a string from *buffer_addr* with length *len* to the file represented by *file_id*.	☑	☑	☑
#REC	S	rec file_id --	--	Sets the record number of file referenced by *file_id* to record *rec* for reading or writing relative files.	☑	☑	☑
FBUF:	I	--	--	Declares a file buffer, used by FILE to build the PAB (Peripheral Access Block) required for file access.	☑	☑	☑
FILE	I		--	Builds a PAB (Peripheral Access Block) in a buffer previously declared with `FBUF:` according to the descriptor string supplied via the stack.	☑	☑	☑
LOAD#5	S	"filename" vdpAddress --	--	Loads an Editor/Assembler Option #5 (raw/program image) file into VDP memory at vdp address *vdpAddress*.	☐	☐	☑
SAVE#5	S	"filename" size saveAddr	--	Saves *size* bytes of VDP memory, starting at VDP memory address *saveAddr* to disk with the given filename.	☐	☐	☑

#CLOSE

In File I/O Words in TurboForth Kernal

Word Name:	#CLOSE
Type:	Standard word
Data Stack Signature:	file_id --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Closes the file with the file id file_id.

Example:	Where a file is opened thus:
	`MYFILE #OPEN`
	the following will close the same file:
	`MYFILE #CLOSE`

Comment:	none
See Also:	None listed

#EOF?

In File I/O Words in TurboForth Kernal

Word Name:	#EOF?
Type:	Standard word
Data Stack Signature:	file_id -- flag
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns TRUE if file file_id is currently positioned at the end of the file.
Example:	none
Comment:	none
See Also:	None listed

#GET

In File I/O Words in TurboForth Kernal

Word Name:	#GET
Type:	Standard word
Data Stack Signature:	buf_addr file_id -- flag
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Reads a record from an external file into a buffer.
Example:	none
Comment:	Reads a line of input from the file specified by *file_id*. The address of an appropriately sized buffer (*buff_addr*) must be supplied. If the read is successful, the buffer at *buff_addr* is filled with the data read from the input device, with the first byte being the length count of the data immediately following it. This can be converted into a address/length pair with COUNT. Returns FALSE if successful or TRUE if not successful. This allows trapping with ABORT" as follows: `MYFILE #GET ABORT" Could not read from file"` If the read fails, IOERR is set to the error code, otherwise it is set to 0.
See Also:	#CLOSE #EOF? #OPEN #PUT #REC

#OPEN

In File I/O Words in TurboForth Kernal

Word Name:	#OPEN
Type:	Standard word
Data Stack Signature:	file_id -- flag
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Opens a file with the file name and attributes specified in the buffer starting at file_id.
Example:	none
Comment:	The buffer (actually a PAB) is set-up with FILE. E.g. `FBUF: SERIAL S" RS232.BA=9600 DV80O" SERIAL FILE SERIAL #OPEN` The above attempts to open the serial port for output as a Display Variable 80 type file. A file buffer called SERIAL is declared (using FBUF:), and this is fed to FILE (along with the string that describes the file). FILE builds the PAB in the buffer named SERIAL. #OPEN leaves a FALSE on the stack if the file was opened successfully. If the file could not be opened, it leaves a TRUE on the stack. This allows easy trapping with ABORT" as shown below: `SERIAL #OPEN ABORT" Can't open file"` In the event of a file error, IOERR can be read to get the DSR error code. If IOERR returns -1 (>FFFF) then this means that no free file IO slots were found. A maximum of 3 open files are supported (2 if block files are also to be used). Note that block files are immediately closed after they are accessed for either reading or writing, so 3 generic file I/O streams are available when no blocks files are being used.
See Also:	#CLOSE #EOF? #GET #PUT #REC FILE

#PUT

Sea

In File I/O Words in TurboForth Kernal

Word Name:	#PUT
Type:	Standard word
Data Stack Signature:	buf_addr len file_id -- flag
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Writes a string from *buffer_addr* with length *len* to the file represented by *file_id*.
Example:	none

Comment:	Returns FALSE if successful, else returns TRUE. This can be trapped with ABORT".
	Example:
	`FBUF: DISK-FILE`
	`S" DSK1.TEST DV80SO" DISK-FILE FILE`
	`: TEST ( -- )`
	`  DISK-FILE #OPEN ABORT" Cannot open disk file for output"`
	`  S" This is a test" DISK-FILE #PUT`
	`  ABORT" Cannot write to disk"`
	`  DISK-FILE #CLOSE`
	`;`

See Also:	#CLOSE #EOF? #GET #OPEN #PUT #REC

#REC

In File I/O Words in TurboForth Kernal

Word Name:	#REC
Type:	Standard word
Data Stack Signature:	rec file_id --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Sets the record number of file referenced by *file_id* to record *rec* for reading or writing relative files.
Example:	none
Comment:	The file should be opened prior to using #REC.
See Also:	#CLOSE #EOF? #GET #OPEN #PUT #REC

FBUF:

In File I/O Words in TurboForth Kernal

Word Name:	FBUF:
Type:	Immediate word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Declares a file buffer, used by FILE to build the PAB (Peripheral Access Block) required for file access.
Example:	none

Comment:	Internally, `FBUF:` ALLOTs 42 bytes starting at `HERE`. Later, `FILE`, as it parses the file-name and file descriptor, will build the PAB in the buffer previously declared with `FBUF:`
	
	E.g.
	
	`FBUF: DISK-FILE`
	`S" DSK1.README DV80SI" DISK-FILE FILE`
	
	Here, a buffer called DISK-FILE is declared. FILE is used to build the PAB for a file on DSK1 called README, declared as a DV80 file, in sequential input mode. Of the 42 bytes allotted, 12 are used for the PAB and internal pointers, leaving 30 characters for the file-name, which allows enough space for hard disk systems with subdirectories etc.
	
	`FBUF:` should *not* be used inside a colon definition.

See Also:	FILE

FILE

In File I/O Words in TurboForth Kernal

Word Name:	FILE
Type:	Immediate word
Data Stack Signature:	
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Builds a PAB (Peripheral Access Block) in a buffer previously declared with `FBUF:` according to the descriptor string supplied via the stack.
Example:	`S" PIO.CR DV80" PRINTER FILE` The above defines a file variable (previously declared with `FBUF:`) called PRINTER which references the PIO device. Subsequent file IO words that wish to send data to the PIO shall use the file variable name to reference it, e.g.: `PRINTER #OPEN DROP` (open PIO and drop success/fail flag) `S" HELLO WORLD" PRINTER #PUT DROP` (write HELLO WORLD to the PIO and drop success/fail flag)
Comment:	The string, which specifies the file name and file characteristics is defined as below. The file-name ***must*** come first followed by at least one space character. After that, the file characteristics can come in any order. *File Access Options* **File Options** / **File Type** F — Fixed record type V — Variable record type D — Display data type L — Internal data type* U — Update file mode O — Output file mode I — Input file mode A — Append file mode S — Sequential file type R — Relative file type *Note that Internal type files require L - this is because I is used to specify INPUT mode. Internally, `FILE` variables build a PAB (peripheral address block) in CPU memory which will be used by `#OPEN` and all File IO words. Word 0 of the reserved memory is used to point to the actual PAB in VDP memory. The VDP location of the PAB is not determined until `#OPEN` is executed.
See Also:	#CLOSE #OPEN FBUF:

LOAD#5

In File I/O Words in TurboForth Kernal

Word Name:	LOAD#5
Type:	Standard word
Data Stack Signature:	"filename" vdpAddress --
Return Stack Signature:	--
Availability:	V1.2
Description:	Loads an Editor/Assembler Option #5 (raw/program image) file into VDP memory at vdp address *vdpAddress*.
Example:	S" DSK1.MYFILE" $1400 LOAD#5
Comment:	Loads a raw binary file (sometimes called program-image) into VDP memory starting at *vdpAddress*. Up to 8192 bytes may be loaded. It is up to the user to transfer the data from VDP to CPU memory (if required) using VMBR or some other means. The loaded data may overwrite file I/O and block buffers, depending on the size of the data. See the VDP Memory Map for more information.
See Also:	VMBR SAVE#5

SAVE#5

In File I/O Words in TurboForth Kernal

Word Name:	SAVE#5
Type:	Standard word
Data Stack Signature:	"filename" size saveAddr
Return Stack Signature:	--
Availability:	V1.2
Description:	Saves *size* bytes of VDP memory, starting at VDP memory address *saveAddr* to disk with the given filename.
Example:	None
Comment:	See VMBW for writing data from CPU memory to VDP memory. Maximum amount of memory that may be saved to one file is 8192 bytes (8K). The data to save may overwrite file I/O and block buffers, depending on the size of the data. See the VDP Memory Map for more information.
See Also:	VMBW LOAD#5

TurboForth

Category 8

Words in category Flow Control Words in glossary TurboForth Kernal

The following words are listed in this category (27 words found): **Word Types:** S = Standard, I = Immediate , IC = Immediate-compiling.

Word Name	Word Type	Data Stack Signature	Return Stack Signature	Description	Availability V1.0	V1.1	V1.2
(+LOOP)	S	n --	-- ret_addr max index | --	Internal assembly-code representation of +LOOP. Compiled by +LOOP.	☑	☑	☑
(DO)	S	max index --	-- ret_addr max index	Internal assembly-code representation of DO. Compiled by DO.	☑	☑	☑
(FOR)	S	count --	-- ret_addr max index	Internal assembly-code representation of FOR. Compiled by FOR.	☑	☑	☐
(LOOP)	S	--	-- ret_addr max index | --	Internal assembly-code represenation of LOOP. Compiled by LOOP.	☑	☑	☑
(NEXT)	S	--	-- ret_addr max index | --	Compiled into a definition by NEXT. Cause the loop index to be evaluated against zero, and the loop to be repeated if the index is not 0.	☑	☑	☐
+LOOP	IC	n --	--	Adds n to the current loop index. If the index equals or exceeds the loop limit, the loop exits, otherwise the loop executes again.	☑	☑	☑
0BRANCH	S	flag --	--	Branches to the address specified in the following cell if flag is FALSE, otherwise takes no action.	☑	☑	☑
AGAIN	IC	--	--	Loops back unconditionally to associated BEGIN.	☑	☑	☑
BEGIN	I	--	--	Marks the beginning of a BEGIN...WHILE...REPEAT loop or a BEGIN...UNTIL loop.	☑	☑	☑
BRANCH	S	--	--	Branches unconditionally to the address in the following cell.	☑	☑	☑
CASE	I	--	--	Marks the beginning of case-by-case processing.	☑	☑	☑
DO	IC	max start --	--	Begins a DO/LOOP or a DO/+LOOP sequence.	☑	☑	☑
ELSE	IC	--	--	Words following ELSE will be executed if the parent IF clause evaluates to FALSE.	☑	☑	☑
ENDCASE	IC	--	--	Ends case-by-case processing. See CASE.	☑	☑	☑
ENDOF	IC	--	--	Marks the end of a particular test case. See CASE.	☑	☑	☑
FOR	IC	count --	--	Begins a FOR/NEXT loop.	☑	☑	☐
I	S	-- index	--	Returns the current index in a DO/LOOP or FOR/NEXT loop.	☑	☑	☑
IF	IC	flag --	--	Executes the words following IF if *flag* evaluates to TRUE. Starts an IF...ELSE...THEN clause.	☑	☑	☑
J	S	-- index	--	Returns the *outer index* when used in a nested DO or FOR loop.	☑	☑	☑
LEAVE	S	--	ret_add max index --	In a DO/LOOP or FOR/NEXT loop, causes the loop to exit. LEAVE should be used inside an IF...THEN block.	☑	☑	☑
LOOP	IC	--	ret_add max index --ret_add index | --	Causes a DO/LOOP to repeat from the word immediately following DO.	☑	☑	☑
NEXT	IC	--	--	Causes a FOR/NEXT loop to repeat from the word following FOR.	☑	☑	☐
OF	IC	--	--	Marks the beginning of a case test. See CASE.	☑	☑	☑
REPEAT	IC	--	--	Loops back unconditionally to associated BEGIN.	☑	☑	☑
THEN	IC	--	--	Marks the end of an IF...ELSE...THEN construct.	☑	☑	☑
UNTIL	IC	flag --	--	Jumps back to associated BEGIN if *flag* evaluates to FALSE. I.e the loop exits if *flag* is TRUE.	☑	☑	☑
WHILE	IC	flag --	--	Enters a WHILE/REPEAT block if flag evaluates to TRUE.	☑	☑	☑

(+LOOP)

In Flow Control Words in TurboForth Kernal

Word Name:	(+LOOP)
Type:	Standard word
Data Stack Signature:	n --
Return Stack Signature:	-- ret_addr max index | --
Availability:	V1.0 V1.1 V1.2
Description:	Internal assembly-code representation of +LOOP. Compiled by +LOOP.
Example:	none
Comment:	none
See Also:	+LOOP

(DO)

In Flow Control Words in TurboForth Kernal

Word Name:	(DO)
Type:	Standard word
Data Stack Signature:	max index --
Return Stack Signature:	-- ret_addr max index
Availability:	V1.0 V1.1 V1.2
Description:	Internal assembly-code representation of DO. Compiled by DO.
Example:	none
Comment:	none
See Also:	DO

(FOR)

In Flow Control Words in TurboForth Kernal

Word Name:	(FOR)
Type:	Standard word
Data Stack Signature:	count --
Return Stack Signature:	-- ret_addr max index
Availability:	V1.0 V1.1
Description:	Internal assembly-code representation of FOR. Compiled by FOR.
Example:	none
Comment:	**Note**: Not present in V1.2. V1.2 uses (DO) and (+LOOP) to produce a loop.
See Also:	FOR

(LOOP)

In Flow Control Words in TurboForth Kernal

Word Name:	(LOOP)
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	-- ret_addr max index | --
Availability:	V1.0 V1.1 V1.2
Description:	Internal assembly-code represenation of LOOP. Compiled by LOOP.
Example:	none
Comment:	none
See Also:	LOOP

(NEXT)

In Flow Control Words in TurboForth Kernal

Word Name:	(NEXT)
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	-- ret_addr max index | --
Availability:	V1.0 V1.1
Description:	Compiled into a definition by NEXT. Cause the loop index to be evaluated against zero, and the loop to be repeated if the index is not 0.
Example:	none
Comment:	**Note**: Not present in V1.2. V1.2 uses (DO) and (+LOOP) to produce a loop.
See Also:	(FOR) NEXT

+LOOP

In Flow Control Words in TurboForth Kernal

Word Name:	+LOOP
Type:	Immediate compiling word
Data Stack Signature:	n --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Adds n to the current loop index. If the index equals or exceeds the loop limit, the loop exits, otherwise the loop executes again.

Example:	`: TEST 100 0 DO I . 2 +LOOP;`
	`TEST`
	(0 2 4 6 8 10 12 14 16 etc)

Comment:	none
See Also:	DO LEAVE LOOP

0BRANCH

In Flow Control Words in TurboForth Kernal

Word Name:	0BRANCH
Type:	Standard word
Data Stack Signature:	flag --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Branches to the address specified in the following cell if flag is FALSE, otherwise takes no action.
Example:	none
Comment:	0BRANCH is compiled into colon definitions by words such as IF, ELSE, WHILE, UNTIL, OF etc. It is very seldomly used in regular Forth colon definitions, though it is often used when defining compiling words (words that compile code into other code).
See Also:	BRANCH

AGAIN

In Flow Control Words in TurboForth Kernal

Word Name:	AGAIN
Type:	Immediate compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Loops back unconditionally to associated BEGIN.

Example:	`: FOREVER 0 BEGIN DUP . 1+ AGAIN ;`
	`FOREVER`
	(0 1 2 3 4 5 6 7 etc...)

Comment:	none
See Also:	BEGIN

BEGIN

In Flow Control Words in TurboForth Kernal

Word Name:	BEGIN
Type:	Immediate word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Marks the beginning of a `BEGIN...WHILE...REPEAT` loop or a `BEGIN...UNTIL` loop.

Example:	`: PRESS-S ( -- )`
	`  CR ." PRESS S"`
	`  BEGIN`
	`  KEY ASCII S = NOT IF`
	`  CR ." I SAID PRESS S!"`
	`  FALSE`
	`  ELSE`
	`  TRUE`
	`  THEN`
	`  UNTIL`
	`  CR ." THANK YOU!" CR`
	`;`

Comment:	none
See Also:	REPEAT UNTIL WHILE

BRANCH

In Flow Control Words in TurboForth Kernal

Word Name:	BRANCH
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Branches unconditionally to the address in the following cell.
Example:	none
Comment:	Generally used in compiling words to compile branches.
See Also:	oBRANCH

CASE

In Flow Control Words in TurboForth Kernal

Word Name:	CASE
Type:	Immediate word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Marks the beginning of case-by-case processing.

Example:	`: TEST ( -- )`
	`  KEY CASE`
	`  32 OF ." SPACE " ENDOF`
	`  42 OF ." STAR " ENDOF`
	`  13 OF ." ENTER " ENDOF`
	`  ." UNKNOWN CASE"`
	`  ENDCASE ;`
	
	`: TEST-CASE ( -- )`
	`  BEGIN TEST BREAK? AGAIN ;`
	
	`TEST-CASE`

Comment:	none
See Also:	ENDCASE ENDOF OF

DO

In Flow Control Words in TurboForth Kernal

Word Name:	DO
Type:	Immediate compiling word
Data Stack Signature:	max start --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Begins a DO/LOOP or a DO/+LOOP sequence.
Example:	`: TEST ( -- ) 100 0 DO I . LOOP ;`
Comment:	The loop begins counting at *start*. The loop repeats when the word LOOP or +LOOP is encountered. The loop repeats from the word following DO. Each iteration through the loop increases the index value (accessible with the word I). When the index value is equal to *max* the loop terminates. The word LEAVE may be used to conditionally exit the loop early.
See Also:	+LOOP I LEAVE LOOP

ELSE

In Flow Control Words in TurboForth Kernal

Word Name:	ELSE
Type:	Immediate compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Words following ELSE will be executed if the parent IF clause evaluates to FALSE.
Example:	`: ODD/EVEN ( n -- ) CR 1 AND IF ." ODD" ELSE ." EVEN" THEN CR ;`
Comment:	none
See Also:	IF THEN

ENDCASE

In Flow Control Words in TurboForth Kernal

Word Name:	ENDCASE
Type:	Immediate compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Ends case-by-case processing. See CASE.
Example:	none
Comment:	none
See Also:	CASE ENDOF OF

ENDOF

In Flow Control Words in TurboForth Kernal

Word Name:	ENDOF
Type:	Immediate compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Marks the end of a particular test case. See CASE.
Example:	none
Comment:	none
See Also:	ENDCASE ENDOF OF

FOR

In Flow Control Words in TurboForth Kernal

Word Name:	FOR
Type:	Immediate compiling word
Data Stack Signature:	count --
Return Stack Signature:	--
Availability:	V1.0 V1.1
Description:	Begins a FOR/NEXT loop.

Example:	`: TEST ( -- ) 10 FOR I . NEXT ;`
	`TEST`
	(displays 10 9 8 7 6 5 4 3 2 1 0)

Comment:	The loop begins counting at *count* and **decreases** by 1 with each iteration. The loop terminates when the loop index (accessible via I) is equal to 0. The word LEAVE may be used to conditionally exit the loop early.
	Note that a FOR loop counts from *count* to 0, whereas a DO loop counts from *count-1* to 0.
	Using a FOR/NEXT loop:
	`: TEST 10 FOR I . NEXT ;`
	`TEST`
	`10 9 8 7 6 5 4 3 2 1 0` (11 values displayed)
	Using a DO/LOOP:
	`: TEST 10 0 DO I . LOOP ;`
	`TEST`
	`0 1 2 3 4 5 6 7 8 9` (10 values displayed)

See Also:	I LEAVE NEXT

I

In Flow Control Words in TurboForth Kernal

Word Name:	I
Type:	Standard word
Data Stack Signature:	-- index
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns the current index in a DO/LOOP or FOR/NEXT loop.
Example:	`: TEST ( -- ) 100 0 DO I . LOOP ;`
Comment:	none
See Also:	Microsoft JET Database Engine error '80040e14' Syntax error (missing operator) in query expression 'ID IN ()'. /lang_ref/view_word.asp, line 123

IF

In Flow Control Words in TurboForth Kernal

Word Name:	IF
Type:	Immediate compiling word
Data Stack Signature:	flag --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Executes the words following `IF` if *flag* evaluates to `TRUE`. Starts an `IF...ELSE...THEN` clause.
Example:	`: ODD/EVEN ( n -- ) CR 1 AND IF ." ODD" ELSE ." EVEN" THEN CR ;`
Comment:	none
See Also:	ELSE THEN

J

In Flow Control Words in TurboForth Kernal

Word Name:	J
Type:	Standard word
Data Stack Signature:	-- index
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns the *outer index* when used in a nested `DO` or `FOR` loop.
Example:	`: TEST ( -- ) 10 0 DO ." Outer loop " I . CR 5 0 DO ." inner loop " J . LOOP CR LOOP ;` In this example the inner loop accesses the index of the outer loop via the word `J`.
Comment:	none
See Also:	DO FOR LOOP NEXT

LEAVE

In Flow Control Words in TurboForth Kernal

Word Name:	LEAVE
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	ret_add max index --
Availability:	V1.0 V1.1 V1.2
Description:	In a DO/LOOP or FOR/NEXT loop, causes the loop to exit. LEAVE should be used inside an IF...THEN block.

Example:	`: TEST ( -- )`
	`  10000 0 DO`
	`  I .`
	`  I 50 = IF ." EXITING EARLY" CR LEAVE THEN`
	`  LOOP`
	`;`
	In this example, the loop is supposed to execute 10,000 times. However, the loop exits when the index is equal to 50.

Comment:	none
See Also:	DO FOR LOOP NEXT

LOOP

In Flow Control Words in TurboForth Kernal

Word Name:	LOOP
Type:	Immediate compiling word
Data Stack Signature:	--
Return Stack Signature:	ret_add max index --ret_add index | --
Availability:	V1.0 V1.1 V1.2
Description:	Causes a DO/LOOP to repeat from the word immediately following DO.
Example:	`: TEST ( -- ) 100 0 DO I . LOOP ;`
Comment:	When the loop index, accessible via I is equal to max (see DO) the loop terminates and execution continues with the word immediatley following DO)
See Also:	+LOOP DO LEAVE

NEXT

In Flow Control Words in TurboForth Kernal

Word Name:	NEXT
Type:	Immediate compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1
Description:	Causes a FOR/NEXT loop to repeat from the word following FOR.
Example:	`: TEST ( -- ) 100 FOR I . NEXT ." Finished!" CR ;`
Comment:	When the loop index, accessible via I is equal to 0 the loop terminates and execution continues with the word immediatley following NEXT.
See Also:	FOR

OF

In Flow Control Words in TurboForth Kernal

Word Name:	OF
Type:	Immediate compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Marks the beginning of a case test. See CASE.
Example:	none
Comment:	none
See Also:	CASE

REPEAT

In Flow Control Words in TurboForth Kernal

Word Name:	REPEAT
Type:	Immediate compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Loops back unconditionally to associated BEGIN.
Example:	none
Comment:	none
See Also:	BEGIN

THEN

In Flow Control Words in TurboForth Kernal

Word Name:	THEN
Type:	Immediate compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Marks the end of an IF...ELSE...THEN construct.
Example:	See IF.
Comment:	none
See Also:	IF

UNTIL

In Flow Control Words in TurboForth Kernal

Word Name:	UNTIL
Type:	Immediate compiling word
Data Stack Signature:	flag --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Jumps back to associated BEGIN if *flag* evaluates to FALSE. I.e the loop exits if *flag* is TRUE.
Example:	See BEGIN.
Comment:	none
See Also:	BEGIN

WHILE

In Flow Control Words in TurboForth Kernal

Word Name:	WHILE
Type:	Immediate compiling word
Data Stack Signature:	flag --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Enters a WHILE/REPEAT block if flag evaluates to TRUE.
Example:	none
Comment:	none
See Also:	BEGIN REPEAT

TurboForth

Category 9

Words in category Graphics Words in glossary TurboForth Kernal

The following words are listed in this category (20 words found): **Word Types:** S = Standard, I = Immediate , IC = Immediate-compiling.

Word Name	Word Type	Data Stack Signature	Return Stack Signature	Description	Availability V1.0	V1.1	V1.2
COINC	S	tolerance sprite1 sprite2 -- flag	--	Checks for coincidence between sprite1 and sprite2. If *both* the horizontal and vertical difference between the two sprites is >= to tolerance then the sprites are not in coincidence with each other and flag shall be false, otherwise it shall be true.	☐	☐	☑
COLOR	S	set fgcolor bgcolor --	--	Sets the character set *set* to the foreground colour *fgcolor* and the background colour *bgcolor*.	☑	☑	☑
DCHAR	S	address len ascii --	--	Defines the character with the code *ascii* using *len* cells of data found at *address*. Normally used in conjunction with DATA.	☑	☑	☑
GCHAR	S	y x -- ascii	--	Returns the ascii code of the character at screen location *y* and *x*.	☑	☑	☑
GMODE	S	mode --	--	Sets the graphics mode to *mode*.	☑	☑	☑
HCHAR	S	y x ascii count --	--	Draws *count ascii* characters on screen starting at *y* & *x*, continuing horizontally and to the right.	☑	☑	☑
MAGNIFY	S	n --	--	Sets the magnification value for sprites to n.	☑	☑	☑
PANEL	S	x y xlength ylength --	--	Logically defines a rectangular panel starting at *y*, *x* extending to *ylength* and *xlength* to be scrolled by SCROLL.	☑	☑	☑
SCREEN	S	color --	--	Sets screen colour to *color*.	☑	☑	☑
SCROLL	S	direction --	--	Scrolls screen as defined by PANEL in direction *direction*.	☑	☑	☑
SMLIST	S	sprite y x --	--	Defines a movement vector for sprite sprite.	☑	☐	☐
SPRCOL	S	sprite color --	--	Sets the colour of sprite *sprite* to colour *color*. Valid colours range is 0 to 15. See SCREEN for colours.	☑	☑	☑
SPRITE	S	sprite y x pattern color --	--	Sets sprite *sprite* to the y location *y*, the x location *x* assigned to pattern *pattern* with colour *color*.	☑	☑	☑
SPRLOC	S	sprite y x --	--	Sets the location of sprite *sprite* to *y* and *x*.	☑	☑	☑
SPRLOC?	S	sprite -- y x	--	Returns the *x* and *y* location of sprite *sprite*.	☑	☑	☑
SPRMOV	S	start_sprite count --	--	Moves the sprites starting at *start_sprite* and continuing for *count* sprites according to their vectors (see SPRVEC).	☑	☑	☑
SPRPAT	S	sprite ascii --	--	Sets the ascii code of sprite *sprite* to *ascii*.	☑	☑	☑
SPRVEC	S	sprite y x --	--	Defines a movement vector for sprite *sprite*.	☐	☐	☑
VCHAR	S	y x ascii count --	--	Draws *count ascii* characters on screen starting at *y* & *x*, continuing downward and to the right.	☑	☑	☑
VWTR	S	value register --	--	Writes the value in *value* to VDP register *register*.	☐	☐	☑

COINC

In Graphics Words in TurboForth Kernal

Word Name:	COINC
Type:	Standard word
Data Stack Signature:	tolerance sprite1 sprite2 -- flag
Return Stack Signature:	--
Availability:	V1.2
Description:	Checks for coincidence between sprite1 and sprite2. If *both* the horizontal and vertical difference between the two sprites is >= to tolerance then the sprites are not in coincidence with each other and flag shall be false, otherwise it shall be true.

Example:	50 value sx 60 value sy 0 value out 0 value hit?
	
	: square (--)
	  \ define a square block for sprite pattern #0
	  data 4 $ffff $ffff $ffff $ffff 256 dchar ;
	
	: ds (--)
	  \ display sprites
	  2 magnify
	  0 50 50 0 3 sprite
	  1 50 60 0 6 sprite ;
	
	: movespr (--)
	  \ use joystick to move sprites. Fire to exit.
	  1 gmode square ds
	  false to out
	  begin
	  0 joyst case
	  1 of true to out endof
	  2 of -1 +to sx endof
	  4 of 1 +to sx endof
	  8 of 1 +to sy endof
	  16 of -1 +to sy endof
	  endcase
	  1 sy sx sprloc
	  8 0 1 coinc dup hit? = if
	  \ collision state hasn't changed. don't update display
	  drop
	  else
	  \ collision state has changed; update display
	  dup to hit?
	  0 0 gotoxy .
	  then
	  out until ;

Comment:	Introduced in TurboForth V1.2.1:4 (September 2015) **Note**: Because checking for sprite coincidence is quite expensive, COINC first tests the VDP sprite collision bit. If the hardware says two sprites are in collision then the test proceeds to see if the nominated sprites are in collision with reference to the tolerance. If the hardware reports no collision then the test does not proceed any further. Distance measuring for tolerance calculation starts at the top left of the sprites.
See Also:	SPRITE SPRLOC SPRLOC? SPRMOV

COLOR

In Graphics Words in TurboForth Kernal

Word Name:	COLOR
Type:	Standard word
Data Stack Signature:	set fgcolor bgcolor --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Sets the character set *set* to the foreground colour *fgcolor* and the background colour *bgcolor*.
Example:	none
Comment:	Only relevant in 32 column mode (i.e. when GMODE is set to 1).
See Also:	GMODE SCREEN

DCHAR

In Graphics Words in TurboForth Kernal

Word Name:	DCHAR
Type:	Standard word
Data Stack Signature:	address len ascii --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Defines the character with the code *ascii* using *len* cells of data found at *address*. Normally used in conjunction with DATA.

Example:	`: TEST ( -- )`
	`  DATA 4 $FFFF $FFFF $FFFF $FFFF 65 DCHAR ;`
	Re-defines the character A as a solid block.

Comment:	none
See Also:	DATA

GCHAR

In Graphics Words in TurboForth Kernal

Word Name:	GCHAR
Type:	Standard word
Data Stack Signature:	y x -- ascii
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns the ascii code of the character at screen location *y* and *x*.
Example:	none
Comment:	none
See Also:	HCHAR VCHAR

GMODE

In Graphics Words in TurboForth Kernal

Word Name:	GMODE
Type:	Standard word
Data Stack Signature:	mode --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Sets the graphics mode to *mode*.
Example:	none
Comment:	Valid mode values are Graphic Modes **Value** / **Description** 0 — 40 column mode 1 — 32 column mode 2 — 80 column mode* *V9938/9958 or F18A video system is required for mode 2.
See Also:	None listed

HCHAR

In Graphics Words in TurboForth Kernal

Word Name:	HCHAR
Type:	Standard word
Data Stack Signature:	y x ascii count --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Draws *count ascii* characters on screen starting at *y* & *x*, continuing horizontally and to the right.
Example:	`0 0 ASCII * 200 HCHAR`
Comment:	none
See Also:	VCHAR

MAGNIFY

In Graphics Words in TurboForth Kernal

Word Name:	MAGNIFY
Type:	Standard word
Data Stack Signature:	n --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Sets the magnification value for sprites to n.
Example:	none
Comment:	Valid values are 0 to 3, as follows: **Value for n** / **Description** 0 — 8x8 pixel sprites, non-magnified 1 — 8x8 pixle sprites, double size 2 — 16x16 pixel sprites, non-magnified 3 — 16x16 pixel sprites, double size
See Also:	SPRITE

PANEL

In Graphics Words in TurboForth Kernal

Word Name:	PANEL
Type:	Standard word
Data Stack Signature:	x y xlength ylength --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Logically defines a rectangular panel starting at *y*, *x* extending to *ylength* and *xlength* to be scrolled by SCROLL.

Example:	`: DELAY ( n -- ) FOR NEXT ;`
	
	`: TEST ( -- )`
	`  WORDS`
	`  10 10 20 20 PANEL`
	`  20 0 DO`
	`  2 SCROLL  1000 DELAY`
	`  LOOP`
	`;`
	
	`TEST`

Comment:	none
See Also:	WRAP SCROLL

SCREEN

In Graphics Words in TurboForth Kernal

Word Name:	SCREEN
Type:	Standard word
Data Stack Signature:	color --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Sets screen colour to *color*.

Example:	`0 GMODE`
	`$F0 SCREEN`
	(sets screen to 40 column mode, with white characters and black screen)

Comment:	In graphics modes 0 and 2, sets screen colour (background colour) and colour of characters as follows: *colour* is an 8 bit value, occupying the lower 8 bits of the top of stack. • The upper 4 bits determine the character colour • The lower 4 bits determine the screen colour In graphics mode 1 (32 column) sets the screen colour only. For setting character colours in mode 1 see `COLOR`.
See Also:	COLOR

SCROLL

In Graphics Words in TurboForth Kernal

Word Name:	SCROLL
Type:	Standard word
Data Stack Signature:	direction --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Scrolls screen as defined by `PANEL` in direction *direction*.
Example:	See `PANEL` for example code.
Comment:	Valid values for direction are: **Value** / **Direction** 0 Left 2 Right 4 Up 6 Down
See Also:	WRAP PANEL

SMLIST

In Graphics Words in TurboForth Kernal

Word Name:	SMLIST
Type:	Standard word
Data Stack Signature:	sprite y x --
Return Stack Signature:	--
Availability:	V1.0
Description:	Defines a movement vector for sprite sprite.
Example:	None
Comment:	When SPRMOV is called, sprite sprite will move y pixels up/down and x pixels left/right. If y is negative sprite will move upwards. If x is negative sprite will move leftwards. **Note**: In TurboForth versions 1.1 and 1.2 this word is called SPRVEC.
See Also:	SPRVEC

SPRCOL

In Graphics Words in TurboForth Kernal

Word Name:	SPRCOL
Type:	Standard word
Data Stack Signature:	sprite color --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Sets the colour of sprite *sprite* to colour *color*. Valid colours range is 0 to 15. See SCREEN for colours.
Example:	none
Comment:	none
See Also:	SCREEN SPRITE

SPRITE

In Graphics Words in TurboForth Kernal

Word Name:	SPRITE
Type:	Standard word
Data Stack Signature:	sprite y x pattern color --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Sets sprite *sprite* to the y location *y*, the x location *x* assigned to pattern *pattern* with colour *color*.

Example:	`HEX`
	`: Face ( -- ) DATA 4 3C42 A581 A599 423C  101 DCHAR ;`
	`DECIMAL`
	`: DoSprite ( -- ) 0 50 60 1 9 SPRITE ;`
	`: Test ( -- ) 1 GMODE  Face DoSprite ;`

Comment:	There are 32 sprites available, numbered from 0 to 31. There are 256 patterns available. Sprite patterns do not share their patterns with the ASCII character set ala Extended Basic, they are independant. Sprite patterns may be defined using DCHAR. When defined using DCHAR, add 256 to the ascii code passed to DCHAR. Sprites are only available in 32-column mode (GMODE=1).
See Also:	GMODE SPRVEC SPRCOL SPRLOC SPRLOC? SPRMOV SPRPAT

SPRLOC

In Graphics Words in TurboForth Kernal

Word Name:	SPRLOC
Type:	Standard word
Data Stack Signature:	sprite y x --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Sets the location of sprite *sprite* to *y* and *x*.
Example:	none
Comment:	none
See Also:	SPRITE SPRLOC?

SPRLOC?

In Graphics Words in TurboForth Kernal

Word Name:	SPRLOC?
Type:	Standard word
Data Stack Signature:	sprite -- y x
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns the *x* and *y* location of sprite *sprite*.
Example:	none
Comment:	none
See Also:	SPRITE SPRLOC

SPRMOV

In Graphics Words in TurboForth Kernal

Word Name:	SPRMOV
Type:	Standard word
Data Stack Signature:	start_sprite count --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Moves the sprites starting at *start_sprite* and continuing for *count* sprites according to their vectors (see SPRVEC).

Example:	`hex`
	`: square ( --) \ define sprite pattern 0 as a solid square`
	`  data 4 $FFFF $FFFF $FFFF $FFFF 100 DCHAR ;`
	`decimal`
	
	`\ define colours:`
	`8 constant red`
	`15 constant white`
	`5 constant blue`
	`11 constant yellow`
	`1 constant black`
	
	`: init ( -- )`
	`  1 gmode  \ 32 column mode`
	`  black screen  \ black screen`
	`  0 magnify  \ single 8x8 sprites`
	`  square  \ define the sprite pattern`
	
	`  \ set up the 4 sprites:`
	`  ( sprite 0) 0 100 100 0 red sprite`
	`  ( sprite 1) 1 110 100 0 white sprite`
	`  ( sprite 2) 2 120 100 0 blue sprite`
	`  ( sprite 3) 3 130 100 0 yellow sprite`
	
	`  \ now define the movement vectors:`
	`  ( sprite 0) 0 0 1 sprvec`
	`  ( sprite 1) 1 0 2 sprvec`
	`  ( sprite 2) 2 0 3 sprvec`
	`  ( sprite 3) 3 0 4 sprvec ;`
	
	`: delay ( n -- ) \ delay loop`
	`  0 do loop ;`
	
	`: moveSprites ( -- ) \ move the sprites 256 times`
	`  init  \ initialise the sprites`
	`  256 0 do`
	`  0 4 sprmov`
	`  500 delay  \ we need a delay otherwise its just a blur!`
	`  loop ;`
	
	`movesprites`

Comment:	none
See Also:	SPRVEC

SPRPAT

In Graphics Words in TurboForth Kernal

Word Name:	SPRPAT
Type:	Standard word
Data Stack Signature:	sprite ascii --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Sets the ascii code of sprite *sprite* to *ascii*.
Example:	none
Comment:	none
See Also:	DCHAR

SPRVEC

In Graphics Words in TurboForth Kernal

Word Name:	SPRVEC
Type:	Standard word
Data Stack Signature:	sprite y x --
Return Stack Signature:	--
Availability:	V1.2
Description:	Defines a movement vector for sprite *sprite*.

Example:	`hex`
	`: square ( --) \ define sprite pattern 0 as a solid square`
	`  data 4 $FFFF $FFFF $FFFF $FFFF 100 DCHAR ;`
	`decimal`
	
	`\ define colours:`
	` 8 constant red`
	`15 constant white`
	` 5 constant blue`
	`11 constant yellow`
	` 1 constant black`
	
	`: init ( -- )`
	`  1 gmode  \ 32 column mode`
	`  black screen  \ black screen`
	`  0 magnify  \ single 8x8 sprites`
	`  square  \ define the sprite pattern`
	
	`  \ set up the 4 sprites:`
	`  ( sprite 0) 0 100 100 0 red sprite`
	`  ( sprite 1) 1 110 100 0 white sprite`
	`  ( sprite 2) 2 120 100 0 blue sprite`
	`  ( sprite 3) 3 130 100 0 yellow sprite`
	
	`  \ now define the movement vectors:`
	`  ( sprite 0) 0 0 1 sprvec`
	`  ( sprite 1) 1 0 2 sprvec`
	`  ( sprite 2) 2 0 3 sprvec`
	`  ( sprite 3) 3 0 4 sprvec ;`
	
	`: delay ( n -- ) \ delay loop`
	`  0 do loop ;`
	
	`: moveSprites ( -- ) \ move the sprites 256 times`
	`  init  \ initialise the sprites`
	`  256 0 do`
	`  0 4 sprmov`
	`  500 delay  \ we need a delay otherwise its just a blur!`
	`  loop ;`

Comment:	When SPRMOV is called, sprite *sprite* will move *y* pixels up/down and *x* pixels left/right. If *y* is negative *sprite* will move upwards. If *x* is negative *sprite* will move leftwards. **Note:** In TurboForth versions 1.0 and 1.1 this word is called SMLIST. It has been renamed in version 1.2 to SPRVEC to better describe its intention/function.
See Also:	SPRMOV SMLIST

VCHAR

In Graphics Words in TurboForth Kernal

Word Name:	VCHAR
Type:	Standard word
Data Stack Signature:	y x ascii count --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Draws *count ascii* characters on screen starting at *y* & *x*, continuing downward and to the right.
Example:	`0 0 ASCII % 768 VCHAR`
Comment:	none
See Also:	HCHAR

VWTR

In Graphics Words in TurboForth Kernal

Word Name:	VWTR
Type:	Standard word
Data Stack Signature:	value register --
Return Stack Signature:	--
Availability:	V1.2
Description:	Writes the value in *value* to VDP register *register*.

Example:	`0 GMODE`
	`$36 7 VWTR`
	The above example changes the screen colour to red, and the text colour to green.

Comment:	Value should be an 8 bit, right justified (i.e. occupying the lower 8 bits) value.
See Also:	GMODE SCREEN

TurboForth

Category 10

Words in category Interpreter Words in glossary TurboForth Kernal

The following words are listed in this category (6 words found): **Word Types:** S = Standard, I = Immediate , IC = Immediate-compiling.

Word Name	Word Type	Data Stack Signature	Return Stack Signature	Description	Availability V1.0	V1.1	V1.2
ABORT	S	--	--	Immediately stops execution, resets data stack, resets return stack and returns to command line. No message is issued.	☑	☑	☑
ABORT"	IC	flag --	--	If flag is true, stops execution, resets stacks, and displays the string. If flag is false then no action is taken.	☑	☑	☑
FORGET	S		--	Removes word “word”, and all words following it, from the dictionary.	☑	☑	☑
INTERPRET	S	--	--	Runs the interpreter/compiler.	☑	☑	☑
MARKER	IC	--	--	Creates a named marker in the dictionary, such that executing the marker causes all words defined *after* it to be erased from the dictionary.	☐	☐	☑
STK?	S	--	--	Checks for stack underflow, and aborts if an underflow is detected, otherwise no action is taken.	☑	☑	☑

ABORT

In Interpreter Words in TurboForth Kernal

Word Name:	ABORT
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Immediately stops execution, resets data stack, resets return stack and returns to command line. No message is issued.

Example:	`: TEST ( -- )`
	`  CR ." Press S to stop" CR`
	`  BEGIN`
	`  42 EMIT`
	`  KEY? ASCII S = IF ABORT THEN`
	`  AGAIN`
	`;`

Comment:	none
See Also:	ABORT"

ABORT"

In Interpreter Words in TurboForth Kernal

Word Name:	ABORT"
Type:	Immediate compiling word
Data Stack Signature:	flag --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	If flag is true, stops execution, resets stacks, and displays the string. If flag is false then no action is taken.

Example:	`: TEST ( -- )`
	`  CR ." Press S to stop" CR`
	`  BEGIN`
	`  42 EMIT`
	`  KEY? ASCII S = ABORT" You pressed S!!"`
	`  AGAIN`
	`;`
	
	`TEST`

Comment:	none
See Also:	ABORT

FORGET

In Interpreter Words in TurboForth Kernal

Word Name:	FORGET
Type:	Standard word
Data Stack Signature:	
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Removes word "word", and all words following it, from the dictionary.

Example:	`: TEST1 1 2 3 ;`
	`: TEST2 4 5 6 ;`
	`FORGET TEST1`
	
	`TEST1`
	(not found)
	`TEST2`
	(not found)

Comment:	Cannot be used in a colon definition. Can be used in a block. The word to erase follows the word `FORGET`. The dictionary pointers `HERE` and `H` are updated, as are the low and high memory pointers `FFAILM` and `FFAIHM`.
See Also:	H FFAIHM FFAILM HERE MARKER

INTERPRET

In Interpreter Words in TurboForth Kernal

Word Name:	INTERPRET
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Runs the interpreter/compiler.
Example:	none
Comment:	none
See Also:	None listed

MARKER

In Interpreter Words in TurboForth Kernal

Word Name:	MARKER
Type:	Immediate compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.2
Description:	Creates a named marker in the dictionary, such that executing the marker causes all words defined *after* it to be erased from the dictionary.

Example:	`: tom ." tom" ;`
	`marker dick`
	`: harry ." harry" ;`
	`dick`
	At this point, tom still exists in the dictionary, but harry (and any other definitions following harry) has been removed from the dictionary.

Comment:	The dictionary pointers `HERE` and `H` are updated, as are the low and high memory pointers `FFAILM` and `FFAIHM`. Cannot be used in a colon definition.
See Also:	FORGET H FFAIHM FFAILM HERE

STK?

In Interpreter Words in TurboForth Kernal

Word Name:	STK?
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Checks for stack underflow, and aborts if an underflow is detected, otherwise no action is taken.
Example:	none
Comment:	none
See Also:	ABORT

TurboForth

Category 11

Words in category Logical Words in glossary TurboForth Kernal

The following words are listed in this category (6 words found):

Word Types: S = Standard, I = Immediate , IC = Immediate-compiling.

Word Name	Word Type	Data Stack Signature	Return Stack Signature	Description	Availability V1.0	V1.1	V1.2
<<	S	a count -- b	--	Shifts *a count* bits to the left, resulting in *b*.	☑	☑	☑
>>	S	a count -- b	--	Shifts *a count* bits to the right, resulting in *b*.	☑	☑	☑
AND	S	a b -- a&&b	--	Replaces *a* and *b* with their logical AND.	☑	☑	☑
NOT	S	a -- ~a	--	Performs a bit-wise inversion (1's complement) on *a*.	☑	☑	☑
OR	S	a b -- a||b	--	Replaces *a* and *b* with their logical OR.	☑	☑	☑
XOR	S	a b -- xor(a,b)	--	Replaces *a* and *b* with their logical XOR (exclusive OR).	☑	☑	☑

<<

In Logical Words in TurboForth Kernal

Word Name:	<<
Type:	Standard word
Data Stack Signature:	a count -- b
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Shifts *a count* bits to the left, resulting in *b*.
Example:	none
Comment:	Bits shifted out of the most significant bit are discarded. Performs an arithmetic shift – i.e. the sign of the word (the most significant bit) is retained.
See Also:	>>

>>

In Logical Words in TurboForth Kernal

Word Name:	>>
Type:	Standard word
Data Stack Signature:	a count -- b
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Shifts *a count* bits to the right, resulting in *b*.
Example:	none
Comment:	Bits shifted out of the least significant bit are discarded. Performs a logical shift – i.e. 0 is shifted in at the most significant bit.
See Also:	<<

AND

In Logical Words in TurboForth Kernal

Word Name:	AND
Type:	Standard word
Data Stack Signature:	a b -- a&&b
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Replaces *a* and *b* with their logical AND.
Example:	none
Comment:	none
See Also:	NOT OR XOR

NOT

In Logical Words in TurboForth Kernal

Word Name:	NOT
Type:	Standard word
Data Stack Signature:	a -- ~a
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Performs a bit-wise inversion (1's complement) on *a*.
Example:	none
Comment:	none
See Also:	AND OR XOR

OR

In Logical Words in TurboForth Kernal

Word Name:	OR
Type:	Standard word
Data Stack Signature:	a b -- a||b
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Replaces *a* and *b* with their logical OR.
Example:	none
Comment:	none
See Also:	AND NOT XOR

XOR

In Logical Words in TurboForth Kernal

Word Name:	XOR
Type:	Standard word
Data Stack Signature:	a b -- xor(a,b)
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Replaces *a* and *b* with their logical XOR (exclusive OR).
Example:	none
Comment:	none
See Also:	AND NOT OR

TurboForth

Category 12

Words in category Math Words in glossary TurboForth Kernal

The following words are listed in this category (21 words found): **Word Types:** S = Standard, I = Immediate , IC = Immediate-compiling.

Word Name	Word Type	Data Stack Signature	Return Stack Signature	Description	Availability V1.0	V1.1	V1.2
-	S	a b -- a-b	--	Subtracts b from a, replacing them with the result.	☑	☑	☑
*	S	a b -- a*b	--	Multiplies a by b, replacing them with the result.	☑	☑	☑
*/	S	n1 n2 n3 – n4	--	*n1* is first multiplied by *n2* producing an intermediate 32-bit result. *n4* is the floor of the quotient of the intermediate 32-bit result divided by the divisor *n3*.	☑	☑	☑
*/MOD	S	n1 n2 n3 -- n4 n5	--	*n1* is first multiplied by *n2* producing an intermediate 32-bit result. *n4* is the remainder and *n5* is the floor of the quotient of the intermediate 32-bit result divided by the divisor *n3*.	☑	☑	☑
/	S	a b -- a/b	--	Divides a by b, replacing them with the result.	☑	☑	☑
/MOD	S	n1 n2 -- n3 n4	--	*n3* is the remainder and *n4* the floor of the quotient of *n1* divided by the divisor *n2*.	☑	☑	☑
+	S	a b -- a+b	--	Sums a and b, replacing them with the result.	☑	☑	☑
1-	S	a -- a-1	--	Subtracts 1 from the topmost cell.	☑	☑	☑
1+	S	a -- a+1	--	Increases the topmost cell value by 1.	☑	☑	☑
2-	S	a -- a-2	--	Subtracts 2 from the topmost cell.	☑	☑	☑
2*	S	a -- a*2	--	Multiplies the topmost cell by 2.	☑	☑	☑
2/	S	a -- a/2	--	Divides the topmost cell by 2.	☑	☑	☑
2+	S	a -- a+2	--	Adds 2 to the topmost cell.	☑	☑	☑
ABS	S	-a -- a	--	Converts topmost cell to positive if negative.	☑	☑	☑
MAX	S	a b -- a|b	--	a and b are removed, to be replaced with the higher of the two.	☑	☑	☑
MIN	S	a b -- a|b	--	a and b are removed, to be replaced with the lower of the two.	☑	☑	☑
MOD	S	a b -- a MOD b	--	Replaces a and b with a mod b.	☑	☑	☑
NEGATE	S	a -- a|-a	--	Negates the topmost cell (positive becomes negative, negative becomes positive).	☑	☑	☑
RND	S	limit --- n	--	Pushes a pseudo random number n which is a number between 0 and limit-1. If the full range (0-65535) is required, use a limit of 0.	☑	☑	☑
UM*	S	u1 u2 -- ud	--	*ud* is the unsigned double (32-bit) product of *u1* times *u2*. All values and arithmetic are unsigned.	☑	☑	☑
UM/MOD	S	ud u1 -- u2 u3	--	*u2* is the remainder and *u3* is the floor of the quotient after dividing *ud* by the divisor *u1*. All values and arithmetic are unsigned. An ambiguous condition results if the divisor is zero or if the quotient lies outside the 16-bit range.	☑	☑	☑

-

In Math Words in TurboForth Kernal

Word Name:	-
Type:	Standard word
Data Stack Signature:	a b -- a-b
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Subtracts b from a, replacing them with the result.
Example:	`100 27 + .`
Comment:	none
See Also:	+

*

In Math Words in TurboForth Kernal

Word Name:	*
Type:	Standard word
Data Stack Signature:	a b -- a*b
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Multiplies a by b, replacing them with the result.
Example:	`10 27 * .`
Comment:	none
See Also:	/

*/

In Math Words in TurboForth Kernal

Word Name:	*/
Type:	Standard word
Data Stack Signature:	n1 n2 n3 – n4
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	*n1* is first multiplied by *n2* producing an intermediate 32-bit result. *n4* is the floor of the quotient of the intermediate 32-bit result divided by the divisor *n3*.
Example:	none
Comment:	The product of n1 and n2 is maintained as an intermediate 32-bit result for greater precision than the otherwise equivalent sequence: n1 n2 * n3 / . An error condition results if the divisor is zero or if the quotient falls outside of the range {-32,768..32,767}.
See Also:	Microsoft JET Database Engine error '80040e14' Syntax error (missing operator) in query expression 'ID IN ()'. /lang_ref/view_word.asp, line 123

*/MOD

In Math Words in TurboForth Kernal

Word Name:	*/MOD
Type:	Standard word
Data Stack Signature:	n1 n2 n3 -- n4 n5
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	*n1* is first multiplied by *n2* producing an intermediate 32-bit result. *n4* is the remainder and *n5* is the floor of the quotient of the intermediate 32-bit result divided by the divisor *n3*.
Example:	none
Comment:	A 32-bit intermediate product is used as for */ . *n4* has the same sign as *n3* or is zero. An ambiguous condition results if the divisor is zero or if the quotient falls outside of the range {-32,768..32,767}.
See Also:	/MOD

/

In Math Words in TurboForth Kernal

Word Name:	/
Type:	Standard word
Data Stack Signature:	a b -- a/b
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Divides a by b, replacing them with the result.
Example:	none
Comment:	none
See Also:	*

/MOD

In Math Words in TurboForth Kernal

Word Name:	/MOD
Type:	Standard word
Data Stack Signature:	n1 n2 -- n3 n4
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	*n3* is the remainder and *n4* the floor of the quotient of *n1* divided by the divisor *n2*.
Example:	none
Comment:	n3 has the same sign as n2 or is zero. An error condition results if the divisor is zero or if the quotient falls outside of the range {-32,768..32,767}.
See Also:	/

+

In Math Words in TurboForth Kernal

Word Name:	+
Type:	Standard word
Data Stack Signature:	a b -- a+b
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Sums a and b, replacing them with the result.
Example:	none
Comment:	none
See Also:	-

1-

In Math Words in TurboForth Kernal

Word Name:	1-
Type:	Standard word
Data Stack Signature:	a -- a-1
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Subtracts 1 from the topmost cell.
Example:	none
Comment:	none
See Also:	1+ 2+ 2-

1+

In Math Words in TurboForth Kernal

Word Name:	1+
Type:	Standard word
Data Stack Signature:	a -- a+1
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Increases the topmost cell value by 1.
Example:	none
Comment:	none
See Also:	2+ 1- 2-

2-

In Math Words in TurboForth Kernal

Word Name:	2-
Type:	Standard word
Data Stack Signature:	a -- a-2
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Subtracts 2 from the topmost cell.
Example:	none
Comment:	none
See Also:	1+ 2+ 2/ 1-

2*

In Math Words in TurboForth Kernal

Word Name:	2*
Type:	Standard word
Data Stack Signature:	a -- a*2
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Multiplies the topmost cell by 2.
Example:	none
Comment:	none
See Also:	2/

2/

In Math Words in TurboForth Kernal

Word Name:	2/
Type:	Standard word
Data Stack Signature:	a -- a/2
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Divides the topmost cell by 2.
Example:	none
Comment:	none
See Also:	2*

ABS

In Math Words in TurboForth Kernal

Word Name:	ABS
Type:	Standard word
Data Stack Signature:	-a -- a
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Converts topmost cell to positive if negative.
Example:	none
Comment:	none
See Also:	NEGATE

MAX

In Math Words in TurboForth Kernal

Word Name:	MAX
Type:	Standard word
Data Stack Signature:	a b -- a|b
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	a and b are removed, to be replaced with the higher of the two.
Example:	none
Comment:	none
See Also:	MIN WITHIN

MIN

In Math Words in TurboForth Kernal

Word Name:	MIN
Type:	Standard word
Data Stack Signature:	a b -- a|b
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	a and b are removed, to be replaced with the lower of the two.
Example:	none
Comment:	none
See Also:	MAX WITHIN

2+

In Math Words in TurboForth Kernal

Word Name:	2+
Type:	Standard word
Data Stack Signature:	a -- a+2
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Adds 2 to the topmost cell.
Example:	none
Comment:	none
See Also:	1+ 2/ 2* 1- 2-

ABS

In Math Words in TurboForth Kernal

Word Name:	ABS
Type:	Standard word
Data Stack Signature:	-a -- a
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Converts topmost cell to positive if negative.
Example:	none
Comment:	none
See Also:	NEGATE

MAX

In Math Words in TurboForth Kernal

Word Name:	MAX
Type:	Standard word
Data Stack Signature:	a b -- a|b
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	a and b are removed, to be replaced with the higher of the two.
Example:	none
Comment:	none
See Also:	MIN WITHIN

MIN

In Math Words in TurboForth Kernal

Word Name:	MIN
Type:	Standard word
Data Stack Signature:	a b -- a|b
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	a and b are removed, to be replaced with the lower of the two.
Example:	none
Comment:	none
See Also:	MAX WITHIN

MOD

In Math Words in TurboForth Kernal

Word Name:	MOD
Type:	Standard word
Data Stack Signature:	a b -- a MOD b
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Replaces a and b with a mod b.
Example:	none
Comment:	none
See Also:	/

NEGATE

In Math Words in TurboForth Kernal

Word Name:	NEGATE
Type:	Standard word
Data Stack Signature:	a -- a|-a
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Negates the topmost cell (positive becomes negative, negative becomes positive).
Example:	none
Comment:	none
See Also:	ABS

RND

In Math Words in TurboForth Kernal

Word Name:	RND
Type:	Standard word
Data Stack Signature:	limit --- n
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Pushes a pseudo random number n which is a number between 0 and limit-1. If the full range (0-65535) is required, use a limit of 0.
Example:	none
Comment:	The seed from which the random numbers derive is determined at power-up.
See Also:	None listed

UM*

In Math Words in TurboForth Kernal

Word Name:	UM*
Type:	Standard word
Data Stack Signature:	u1 u2 -- ud
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	*ud* is the unsigned double (32-bit) product of *u1* times *u2*. All values and arithmetic are unsigned.
Example:	none
Comment:	none
See Also:	*

UM/MOD

In Math Words in TurboForth Kernal

Word Name:	UM/MOD
Type:	Standard word
Data Stack Signature:	ud u1 -- u2 u3
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	*u2* is the remainder and *u3* is the floor of the quotient after dividing *ud* by the divisor *u1*. All values and arithmetic are unsigned. An ambiguous condition results if the divisor is zero or if the quotient lies outside the 16-bit range.
Example:	none
Comment:	none
See Also:	/MOD

TurboForth

Category 13

Words in category Memory Access Words in glossary TurboForth Kernal

The following words are listed in this category (23 words found): **Word Types:** S = Standard, I = Immediate , IC = Immediate-compiling.

Word Name	Word Type	Data Stack Signature	Return Stack Signature	Description	Availability V1.0	V1.1	V1.2
!	S	value address --	--	Writes *value* to the cell at *address*.	☑	☑	☑
@	S	address -- value	--	Pushes the value read from *address* to the stack.	☑	☑	☑
@++	S	address - address+2 value	--	The cell at *address* is read and the value pushed to the stack, above the address. Address is incremented by 2, ready for @++ to be called again. Note, TurboForth V1.0 has a different stack signature. Please see comments.	☑	☑	☑
+!	S	value address --	--	Read the cell at *address*, adds *value* to it and stores the result at *address*.	☑	☑	☑
>MAP	S	bank address --	--	Maps bank number *bank* into memory at address *address*. SAMS 1MB memory expansion required.	☑	☑	☑
0!	S	address --	--	Store 0 to the cell at *address*.	☑	☑	☑
C!	S	char address --	--	Store lower 8 bits of *char* to *address*. (Byte write)	☑	☑	☑
C@	S	address -- n	--	Read byte at *address*. The upper 8 bits of the cell on the data stack are 0. The lower 8 bits contain the value read from the memory address.	☑	☑	☑
CELLS	S	a -- a*2	--	Returns the number of bytes needed for *a* cells (a*2).	☑	☑	☑
CHARS	S	a -- a	--	Returns *a*. Included for compatibility and code readability.	☑	☑	☑
CMOVE	S	src_addr dest_addr count --	--	Copies *count* bytes from *src_addr* to *dest_addr*.	☑	☑	☑
CMOVE>	S	src_addr dest_addr count --	--	Copies *count* bytes from *src_addr* to *dest_addr*.	☑	☑	☑
COPYW	S	src_addr dest_addr count --	--	Copies count cells (16-bits) from src_addr to dest_addr.	☑	☑	☑
FILL	S	address count value --	--	Fills count bytes of memory with value, starting at address.	☑	☑	☑
HFREE	S	-- n	--	Returns available memory (in bytes) in high memory.	☑	☑	☑
LFREE	S	-- n	--	Returns available memory (in bytes) in low memory.	☑	☑	☑
MEM	S	--	--	Reports the number of free bytes in low memory, number of free bytes in high memory, and total number of free bytes respectively to the screen.	☐	☐	☑
V!	S	value address --	--	Stores 8 bit value *value* to VDP address *address*.	☑	☑	☑
V@	S	address -- value	--	Fetches *value* from VDP address *address*. Places the 8-bit value read from VDP memory on the stack as a 16-bit right justified cell.	☑	☑	☑
V2!	S	value vdpAddress --	--	Writes the 16-bit value *value* to *vdpAddress*.	☐	☐	☑
V2@	S	vdpAddress -- n	--	Reads the 16-bit value from address vdpAddress and pushes it to the stack as a single 16-bit cell.	☐	☐	☑
VMBR	S	vdp_address cpu_address count --	--	VDP Multiple Byte Read. Copies *count* bytes starting at *vdp_address* to CPU memory starting at *cpu_address*.	☑	☑	☑
VMBW	S	vdp_address cpu_address count --	--	VDP Multiple Byte Write. Copies *count* bytes starting at CPU address *cpu_address* to vdp memory starting at *vdp_address*.	☑	☑	☑

!

In Memory Access Words in TurboForth Kernal

Word Name:	!
Type:	Standard word
Data Stack Signature:	value address --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Writes *value* to the cell at *address*.
Example:	`99 $B000 !`
Comment:	none
See Also:	@ VARIABLE

@

In Memory Access Words in TurboForth Kernal

Word Name:	@
Type:	Standard word
Data Stack Signature:	address -- value
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Pushes the value read from *address* to the stack.
Example:	`$B000 @`
Comment:	none
See Also:	! VARIABLE

@++

In Memory Access Words in TurboForth Kernal

Word Name:	@++
Type:	Standard word
Data Stack Signature:	address - address+2 value
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	The cell at *address* is read and the value pushed to the stack, above the address. Address is incremented by 2, ready for `@++` to be called again. Note, TurboForth V1.0 has a different stack signature. Please see comments.
Example:	None.
Comment:	Note: In version 1.0, the output stack signature is reversed. **Version** / **Stack Signature** 1.0 — address -- value address+2 1.1 — address -- address+2 value 1.2 — address -- address+2 value
See Also:	@

+!

In Memory Access Words in TurboForth Kernal

Word Name:	+!
Type:	Standard word
Data Stack Signature:	value address --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Read the cell at *address*, adds *value* to it and stores the result at *address*.
Example:	none
Comment:	none
See Also:	! @ VARIABLE

>MAP

In Memory Access Words in TurboForth Kernal

Word Name:	>MAP
Type:	Standard word
Data Stack Signature:	bank address --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Maps bank number *bank* into memory at address *address*. SAMS 1MB memory expansion required.
Example:	none
Comment:	*Address* should be a valid address on a 4K boundary (e.g. >2000, >3000, >A000, >B000, >C000, >D000, >E000 or >F000). *Bank* should be a number between 0 and >FF. **Important Notes:** • When a SAMS memory expansion card is fitted, the 32K of CPU RAM is actually taken from the SAMS memory. At startup TurboForth reserves the following banks of SAMS memory for "standard" 32K RAM:

Bank	Description	Notes
F8	4K @ >2000	8K Lower Memory Expansion
F9	4K @ >3000	
FA	4K @ >A000	24K Upper Memory Expansion
FB	4K @ >B000	
FC	4K @ >C000	
FD	4K @ >D000	
FE	4K @ >E000	
FF	4K @ >F000	

- TurboForth reserves approximately ~770 bytes for its own use starting at >A000, therefore extreme care should be taken when paging bank FA out of >A000;
- As can be seen from the above table, TurboForth assumes a 1024K **S**AMS memory card; TurboForth is not compatible with 256K AMS cards.

See Also:	None listed

o!

In Memory Access Words in TurboForth Kernal

Word Name:	o!
Type:	Standard word
Data Stack Signature:	address --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Store 0 to the cell at *address*.
Example:	none
Comment:	none
See Also:	!

C!

In Memory Access Words in TurboForth Kernal

Word Name:	C!
Type:	Standard word
Data Stack Signature:	char address --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Store lower 8 bits of *char* to *address*. (Byte write)
Example:	none
Comment:	none
See Also:	C@

C@

In Memory Access Words in TurboForth Kernal

Word Name:	C@
Type:	Standard word
Data Stack Signature:	address -- n
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Read byte at *address*. The upper 8 bits of the cell on the data stack are 0. The lower 8 bits contain the value read from the memory address.

Example:	`99 $B001 C!`
	`$B001 C@ .`
	
	Displays 99. Note that the value pushed to the stack is 16-bits wide, the most significant 8 bits (the high byte) of the stack cell are 0, the lower 8 bits contain the value read from location $B001.

Comment:	none
See Also:	C!

CELLS

In Memory Access Words in TurboForth Kernal

Word Name:	CELLS
Type:	Standard word
Data Stack Signature:	a -- a*2
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns the number of bytes needed for *a* cells (a*2).
Example:	none
Comment:	none
See Also:	2*

CHARS

In Memory Access Words in TurboForth Kernal

Word Name:	CHARS
Type:	Standard word
Data Stack Signature:	a -- a
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns *a*. Included for compatibility and code readability.
Example:	none
Comment:	none
See Also:	CELLS

CMOVE

In Memory Access Words in TurboForth Kernal

Word Name:	CMOVE
Type:	Standard word
Data Stack Signature:	src_addr dest_addr count --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Copies *count* bytes from *src_addr* to *dest_addr*.
Example:	none
Comment:	none
See Also:	CMOVE> COPYW FILL

CMOVE>

In Memory Access Words in TurboForth Kernal

Word Name:	CMOVE>
Type:	Standard word
Data Stack Signature:	src_addr dest_addr count --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Copies *count* bytes from *src_addr* to *dest_addr*.
Example:	none
Comment:	Begins copying at the top of the range. Suitable for use when the source and destination memory areas overlap.
See Also:	CMOVE COPYW FILL

COPYW

In Memory Access Words in TurboForth Kernal

Word Name:	COPYW
Type:	Standard word
Data Stack Signature:	src_addr dest_addr count --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Copies count cells (16-bits) from src_addr to dest_addr.
Example:	none
Comment:	none
See Also:	CMOVE CMOVE> FILL

FILL

In Memory Access Words in TurboForth Kernal

Word Name:	FILL
Type:	Standard word
Data Stack Signature:	address count value --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Fills count bytes of memory with value, starting at address.
Example:	none
Comment:	none
See Also:	CMOVE CMOVE> COPYW

HFREE

In Memory Access Words in TurboForth Kernal

Word Name:	HFREE
Type:	Standard word
Data Stack Signature:	-- n
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns available memory (in bytes) in high memory.
Example:	none
Comment:	none
See Also:	LFREE FFAIHM FFAILM

LFREE

In Memory Access Words in TurboForth Kernal

Word Name:	LFREE
Type:	Standard word
Data Stack Signature:	-- n
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns available memory (in bytes) in low memory.
Example:	none
Comment:	none
See Also:	HFREE FFAIHM FFAILM

MEM

In Memory Access Words in TurboForth Kernal

Word Name:	MEM
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.2
Description:	Reports the number of free bytes in low memory, number of free bytes in high memory, and total number of free bytes respectively to the screen.
Example:	none
Comment:	none
See Also:	HFREE LFREE FFAIHM FFAILM

V!

In Memory Access Words in TurboForth Kernal

Word Name:	V!
Type:	Standard word
Data Stack Signature:	value address --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Stores 8 bit value *value* to VDP address *address*.
Example:	none
Comment:	none
See Also:	C!

V@

In Memory Access Words in TurboForth Kernal

Word Name:	V@
Type:	Standard word
Data Stack Signature:	address -- value
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Fetches *value* from VDP address *address*. Places the 8-bit value read from VDP memory on the stack as a 16-bit right justified cell.
Example:	none
Comment:	none
See Also:	C@

V2!

In Memory Access Words in TurboForth Kernal

Word Name:	V2!
Type:	Standard word
Data Stack Signature:	value vdpAddress --
Return Stack Signature:	--
Availability:	V1.2
Description:	Writes the 16-bit value *value* to *vdpAddress*.
Example:	None
Comment:	**Note:** It is not neccessary for vdpAddress to be aligned on a 16-bit boundary. Introduced in TurboForth V1.2.1:4 (September 2015)
See Also:	V! V@ V2@

V2@

In Memory Access Words in TurboForth Kernal

Word Name:	V2@
Type:	Standard word
Data Stack Signature:	vdpAddress -- n
Return Stack Signature:	--
Availability:	V1.2
Description:	Reads the 16-bit value from address vdpAddress and pushes it to the stack as a single 16-bit cell.
Example:	None
Comment:	**Note:** It is not necessary for vdpAddress to be word aligned. Introduced in TurboForth V1.2.1:4 (September 2015)
See Also:	V! V@ V2!

VMBR

In Memory Access Words in TurboForth Kernal

Word Name:	VMBR
Type:	Standard word
Data Stack Signature:	vdp_address cpu_address count --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	VDP Multiple Byte Read. Copies *count* bytes starting at *vdp_address* to CPU memory starting at *cpu_address*.
Example:	none
Comment:	No action is taken if *count*=0.
See Also:	V! V@ VMBW

VMBW

In Memory Access Words in TurboForth Kernal

Word Name:	VMBW
Type:	Standard word
Data Stack Signature:	vdp_address cpu_address count --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	VDP Multiple Byte Write. Copies *count* bytes starting at CPU address *cpu_address* to vdp memory starting at *vdp_address*.
Example:	No action is taken if *count*=0.
Comment:	none
See Also:	V! V@ VMBR

TurboForth

Category 14

Words in category Miscellaneous in glossary TurboForth Kernal

Search: Search

The following words are listed in this category (9 words found):

Word Types: S = Standard, I = Immediate , IC = Immediate-compiling.

Word Name	Word Type	Data Stack Signature	Return Stack Signature	Description	Availability V1.0	V1.1	V1.2
BL	S	-- 32	--	Pushes the value 32, which represents a space character, or a “Blank”. Commonly used in conjunction with WORD to get a word delimited by a blank.	☑	☑	☑
BYE	S	--	--	Immediately resets the computer back the master title screen. Any un-flushed files are not flushed. Any open files are not closed.	☑	☑	☑
COLD	S	--	--	Immediately reboots TurboForth. Any un-flushed files are not flushed prior to reset. Any open files are not closed.	☑	☑	☑
DATA	IC	-- address count	--	Defines a list of numeric data. At runtime, the address of the data, and the number of items in the data list are pushed to the stack.	☑	☑	☑
DECIMAL	S	--	--	Sets the number base to decimal by writing 10 (decimal) to BASE.	☑	☑	☑
EDIT	S	block# --	--	Invokes the built-in editor, loading disk block *block#* into the editor.	☑	☑	☑
EXIT	S	--	--	Exits a word immediately, returning control to the calling word.	☑	☑	☑
HEX	S	--	--	Sets the number base to hexadecimal to by writing 16 (decimal) to BASE.	☑	☑	☑
QUIT	S	--	--	Clears the return stack, sets interpret state and inokes the interpreter.	☑	☑	☑

BL

In Miscellaneous in TurboForth Kernal

Word Name:	BL
Type:	Standard word
Data Stack Signature:	-- 32
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Pushes the value 32, which represents a space character, or a "Blank". Commonly used in conjunction with `WORD` to get a word delimited by a blank.
Example:	`BL WORD`
Comment:	none
See Also:	WORD

BYE

In Miscellaneous in TurboForth Kernal

Word Name:	BYE
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Immediately resets the computer back the master title screen. Any un-flushed files are not flushed. Any open files are not closed.
Example:	none
Comment:	none
See Also:	COLD

COLD

In Miscellaneous in TurboForth Kernal

Word Name:	COLD
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Immediately reboots TurboForth. Any un-flushed files are not flushed prior to reset. Any open files are not closed.
Example:	none
Comment:	none
See Also:	BYE

DATA

In Miscellaneous in TurboForth Kernal

Word Name:	DATA
Type:	Immediate compiling word
Data Stack Signature:	-- address count
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Defines a list of numeric data. At runtime, the address of the data, and the number of items in the data list are pushed to the stack.
Example:	`: TEST ( -- address count) DATA 5 100 200 300 400 500 ;` The above causes 6 cells to be compiled into the word (the length (5) and the five data items). When TEST is subsequently executed two values are pushed to the stack. The topmost stack item (*count*) is the number of data items in the list. The next item on the stack (*address*) is the address in CPU memory of the first item of data.
Comment:	At compile time, compiles the numeric data following it into the colon definition. At run-time, DATA pushes the address of the data, and item count (in cells) to the stack, and continues execution with the word immediately following the data list. The first data parameter should be the number of data items in the list. At run-time, DATA will push the address of the first data item (100) and the count (5) to the stack and continue execution from the first word following the data list. DATA should only be used inside a colon definition. Note: Care should be taken when used on the command line. When used on the command line, a line of data should be no longer than 80 characters. This limitation does apply when compiling from blocks. Note: a considerably enhanced version of DATA is available on the tools disk. Usage Information and source code is given here. It is suggested that where possible, this new enhanced version is used.
See Also:	DCHAR

DECIMAL

In Miscellaneous in TurboForth Kernal

Word Name:	DECIMAL
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Sets the number base to decimal by writing 10 (decimal) to BASE.
Example:	none
Comment:	none
See Also:	HEX BASE

EDIT

In Miscellaneous in TurboForth Kernal

Word Name:	EDIT
Type:	Standard word
Data Stack Signature:	block# --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Invokes the built-in editor, loading disk block *block#* into the editor.
Example:	none
Comment:	If the block cannot be loaded the system returns to the command line.
See Also:	LIST

EXIT

In Miscellaneous in TurboForth Kernal

Word Name:	EXIT
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Exits a word immediately, returning control to the calling word.
Example:	none
Comment:	Do not use inside a loop without first removing the loop control parameters from the return stack (use LEAVE instead).
See Also:	QUIT LEAVE ABORT ABORT"

HEX

In Miscellaneous in TurboForth Kernal

Word Name:	HEX
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Sets the number base to hexadecimal to by writing 16 (decimal) to BASE.
Example:	none
Comment:	none
See Also:	DECIMAL BASE

QUIT

In Miscellaneous in TurboForth Kernal

Word Name:	QUIT
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Clears the return stack, sets interpret state and inokes the interpreter.
Example:	none
Comment:	none
See Also:	EXIT ABORT ABORT"

TurboForth

Category 15

Words in category Parsing Words in glossary TurboForth Kernal

Search: Search

The following words are listed in this category (10 words found):

Word Types: S = Standard, I = Immediate , IC = Immediate-compiling.

Word Name	Word Type	Data Stack Signature	Return Stack Signature	Description	Availability V1.0	V1.1	V1.2
(	I	--	--	Begins a comment. End with a) character. The comment is not compiled into colon definitions.	☑	☑	☑
.(	I	--	--	Begins a displayed comment.	☑	☑	☑
\	I	--	--	Trailing comment. All text after \ will be ignored. The comment is not compiled into colon definitions.	☑	☑	☑
>BODY	S	cfa -- body_address	--	Given a *CFA* on the stack, >BODY returns the address of the body of the word. The "body" contains the "payload" of the word.	☑	☑	☑
>CFA	S	da -- cfa	--	Converts dictionary address *da* to code field address *cfa*.	☑	☑	☑
>LINK	S	cfa -- lfa	--	Converts code field address *cfa* to the associated link field address *lfa*.	☐	☐	☑
EXPECT	S	vdp_address length --	--	Reads characters from the current input device (either the keyboard, or a disk block).	☑	☑	☑
FIND	S	address length -- xt type	--	Searches the dictionary for the string stored at *address* with length of *length* bytes.	☑	☑	☑
NUMBER	S	address length -- value flag	--	Attempts to convert the string at *address* with length *length* into a number.	☑	☑	☑
WORD	S	delimiter -- address length	--	Identifies words from a string of text in the Terminal Input Buffer (which is in VDP memory).	☑	☑	☑

(

In Parsing Words in TurboForth Kernal

Word Name:	(
Type:	Immediate word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Begins a comment. End with a) character. The comment is not compiled into colon definitions.
Example:	none
Comment:	none
See Also:	.(

.(

In Parsing Words in TurboForth Kernal

Word Name:	.(
Type:	Immediate word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Begins a displayed comment.
Example:	none
Comment:	Only valid in a block. End with a) character. The comment will be displayed to the screen as the block loads. Useful in a block to provide commentary as a block loads (loading progress etc.). The comment is not compiled into colon definitions.
See Also:	(

\

In Parsing Words in TurboForth Kernal

Word Name:	\
Type:	Immediate word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Trailing comment. All text after \ will be ignored. The comment is not compiled into colon definitions.
Example:	none
Comment:	none
See Also:	None listed

>BODY

In Parsing Words in TurboForth Kernal

Word Name:	>BODY
Type:	Standard word
Data Stack Signature:	cfa -- body_address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Given a *CFA* on the stack, >BODY returns the address of the body of the word. The "body" contains the "payload" of the word.
Example:	none
Comment:	Only applies to words created with CREATE (i.e. VARIABLE, CONSTANT, VALUE). It is not meaningful to apply >BODY to words that are not children of CREATE. >BODY may also be applied to words that utilise DOES> to carry out their work.
See Also:	CREATE DOES>

>CFA

In Parsing Words in TurboForth Kernal

Word Name:	>CFA
Type:	Standard word
Data Stack Signature:	da -- cfa
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Converts dictionary address *da* to code field address *cfa*.
Example:	none
Comment:	none
See Also:	>LINK

>LINK

In Parsing Words in TurboForth Kernal

Word Name:	>LINK
Type:	Standard word
Data Stack Signature:	cfa -- lfa
Return Stack Signature:	--
Availability:	V1.2
Description:	Converts code field address *cfa* to the associated link field address *lfa*.
Example:	none
Comment:	The link field address is the first field of any dictionary entry. (Note: this word was called >DFA in previous versions).
See Also:	>BODY >CFA

EXPECT

In Parsing Words in TurboForth Kernal

Word Name:	EXPECT
Type:	Standard word
Data Stack Signature:	vdp_address length --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Reads characters from the current input device (either the keyboard, or a disk block).

Example:	**Note:** this example is for version 1.2
	`: test ( -- )`
	`  tib @ 80 expect \ get up to 80 characters of text into TIB in VDP RAM`
	`  begin bl word dup while cr type repeat`
	`  2drop ;`
	`test fred bloggs was ere`
	Output:
	fred
	bloggs
	was
	ere
	Note: each invocation of WORD copies a word from the Termial Input Buffer (TIB) in VDP memory to a word buffer in CPU memory; thus the address returned by WORD and passed to TYPE in the above example is a CPU memory address.

Comment:	Receive *length* (up to a maximum of C/L) characters and store each received character into VDP memory. The transfer begins at *vdp_address*, proceeding towards higher addresses one byte per character until either a "return" is received or until *length* or C/L characters have been transferred. No more than *length* or C/L characters will be stored. The "return" is not stored into memory. No characters are received or transferred if length is zero. On exit, SPAN is set to the number of characters actually received and >IN shall be reset to 0.
See Also:	WORD #TIB >IN C/L SPAN TIB

FIND

In Parsing Words in TurboForth Kernal

Word Name:	FIND
Type:	Standard word
Data Stack Signature:	address length -- xt type
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Searches the dictionary for the string stored at *address* with length of *length* bytes.
Example:	none
Comment:	Returns the execution token (xt) or 0 if the word is not found in the dictionary. *Type* describes the type of the word as follows: Type

Value	Description
0	The word was not found
-1	Standard (non-immediate) word
1	Immediate word

See Also:	>BODY >CFA >LINK WORD #>

NUMBER

In Parsing Words in TurboForth Kernal

Word Name:	NUMBER
Type:	Standard word
Data Stack Signature:	address length -- value flag
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Attempts to convert the string at *address* with length *length* into a number.
Example:	`S" 150" NUMBER DROP .`
Comment:	If fully successful the number (*value*) is placed onto the stack and *flag* will be 0. If it fails (for example contains an illegal character) then a partial number will be placed onto the stack for *value* (the value computed up until the failure) and *flag* will be >0. Thus, if flag>0 the string failed to parse fully as a number. A minus sign is permitted for negative numbers. `NUMBER` parses numbers from the input stream in the current number base, as determined by `BASE`. Eg. If `BASE`=16 then digits 0-9 and A-F are considered legal and will be parsed properly. A facility also exists called 'quick hex' that allows a number to be entered in base 16, by placing a $ symbol at the beginning of the string. This avoids the need to change `BASE` to enter a hex number. E.g. instead of `HEX FEED DECIMAL` you can simply type `$FEED`. The number will be parsed as a hexadecimal number without the need to change BASE. The same facility also exists for binary numbers: Simply precede the number with a % symbol. E.g. `%1001` = 9 decimal. The numbers returned are (by default) singles (16 bits). `NUMBER` can also return a double (32-bit (2 stack cells)) value by including a period in the number string. E.g. 100. 1.00 10.0 .100 will all return 100 decimal (assuming `BASE`=10) as a double, occupying two stack cells. The stack signature when returning a double value is: *address length -- value_lsb value_msb flag*. The various facilities can be mixed. For example, -$F. means -15 as a double. - $ and . can be specified in any order. However, $ or % if required, should be specified before any number digits. - and . can be placed anywhere in the string.
See Also:	WORD BASE

WORD

In Parsing Words in TurboForth Kernal

Word Name:	WORD
Type:	Standard word
Data Stack Signature:	delimiter -- address length
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Identifies words from a string of text in the Terminal Input Buffer (which is in VDP memory).

Example:	`: test ( -- )`
	`  tib @ 80 expect \ get up to 80 characters of text into TIB in VDP RAM`
	`  begin bl word dup while cr type repeat`
	`  2drop ;`
	
	`test fred bloggs was ere`
	
	Output:
	
	fred
	bloggs
	was
	ere

Comment:	**Versions 1.0 & 1.1** Moves through the input buffer defined by `TIB` returns the address and length of a word identified by the ascii value *delimiter*. Updates `>IN` as `WORD` moves through the buffer. If no word is found, or the end of the buffer (as defined by `C/L`) is reached then 0 0 is returned. • *address* shall be a VDP address if `BLK`>0 when `WORD` is called; • *length* shall be the lenth of the identified word, in bytes. **Version 1.2** Moves through the input buffer, which is in VDP memory, and pointed to by TIB, and copies a single blank delimited word to the word buffer in CPU RAM. • *address* shall be a CPU RAM address; • *length* shall be the lenth of the identified word, in bytes.
See Also:	>IN C/L TIB

TurboForth

Category 16

Words in category Return Stack Words in glossary TurboForth Kernal

Search: Search

The following words are listed in this category (5 words found):

Word Types: S = Standard, I = Immediate , IC = Immediate-compiling.

Word Name	Word Type	Data Stack Signature	Return Stack Signature	Description	Availability V1.0	V1.1	V1.2
>R	S	n --	-- n	Moves the top cell from the data stack to the return stack.	☑	☑	☑
R@	S	-- n	n -- n	Copies the top cell from the return stack to the data stack (i.e. does not remove it).	☑	☑	☑
R>	S	-- n	n --	Moves the top cell from the return stack to the data stack.	☑	☑	☑
RP!	S	-- address	--	Sets the return stack pointer.	☑	☑	☐
RP@	S	-- address	--	Pushes the address of the return stack pointer variable.	☑	☑	☑

>R

In Return Stack Words in TurboForth Kernal

Word Name:	>R
Type:	Standard word
Data Stack Signature:	n --
Return Stack Signature:	-- n
Availability:	V1.0 V1.1 V1.2
Description:	Moves the top cell from the data stack to the return stack.
Example:	none
Comment:	Care should be taken to ensure the return stack is in the same state upon leaving a colon definition as it was when entering. DO/LOOP and FOR/NEXT use the return stack to hold loop parameters, therefore, if pushing data temporarily to the return stack during a loop, the return stack should be restore before a LOOP, +LOOP or NEXT is executed.
See Also:	R> R@

R@

In Return Stack Words in TurboForth Kernal

Word Name:	R@
Type:	Standard word
Data Stack Signature:	-- n
Return Stack Signature:	n -- n
Availability:	V1.0 V1.1 V1.2
Description:	Copies the top cell from the return stack to the data stack (i.e. does not remove it).
Example:	none
Comment:	none
See Also:	>R R>

R>

In Return Stack Words in TurboForth Kernal

Word Name:	R>
Type:	Standard word
Data Stack Signature:	-- n
Return Stack Signature:	n --
Availability:	V1.0 V1.1 V1.2
Description:	Moves the top cell from the return stack to the data stack.
Example:	none
Comment:	Care should be taken to ensure the return stack is in the same state upon leaving a colon definition as it was when entering. DO/LOOP and FOR/NEXT use the return stack to hold loop parameters, therefore, if pushing data temporarily to the return stack during a loop, the return stack should be restore before a LOOP, +LOOP or NEXT is executed.
See Also:	>R R@

RP!

In Return Stack Words in TurboForth Kernal

Word Name:	RP!
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1
Description:	Sets the return stack pointer.
Example:	none

Comment:	Sets the return stack pointer to a new address.
	
	Care should be taken when setting the return stack address. Generally, the contents of the old return stack should be copied to the new location before setting the return stack address, or a crash will almost certainly ensue.
	
	Note: V1.2 does not contain RP! however, it is possible to set the return stack address with the following assembly code:
	
	`ASM: RP! ( addr -- )`
	`  *SP+ R3 ** MOV, \ load R3 (return stack pointer) with addr`
	`;ASM`

See Also:	RP@

RP@

In Return Stack Words in TurboForth Kernal

Word Name:	RP@
Type:	Standard word
Data Stack Signature:	-- address
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Pushes the address of the return stack pointer variable.
Example:	`RP@ $.`
Comment:	Points to the *current address* of the return stack pointer.
See Also:	RP! R0

TurboForth

Category 17

Words in category Sound Words in glossary TurboForth Kernal

The following words are listed in this category (1 words found): **Word Types:** S = Standard, I = Immediate , IC = Immediate-compiling.

Word Name	Word Type	Data Stack Signature	Return Stack Signature	Description	Availability V1.0	V1.1	V1.2
SOUND	S	pitch vol ch#	--	Sets sound channel *ch#* to the pitch *pitch* with volume *volume*.	☑	☑	☑

SOUND

In Sound Words in TurboForth Kernal

Word Name:	SOUND
Type:	Standard word
Data Stack Signature:	pitch vol ch#
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Sets sound channel *ch#* to the pitch *pitch* with volume *volume*.

Example:	`: BusyNoise ( -- )`
	`  50 0 DO 50`
	`  15 0 DO`
	`  DUP I 0 SOUND 10 +`
	`  LOOP DROP`
	`  LOOP`
	`  ( turn sound off:) 0 15 0 SOUND`
	`;`

Comment:	Suitable values for pitch can be found in the editor assembler manual on page 318 (note, it is not necessary to adjust the pitch values for each channel as described in the editor assembler book, just use the values given for channel 0 – SOUND modifies the pitch internally for each channel). Valid *vol* values are 0 to 15 (0=loudest). Valid *ch#* values are 0 to 3 (3 is the noise channel). For the noise channel, the pitch field is used to describe the noise. Bits 15 and 14 (15=LSB) select the shift rate, and bit 13 selects either periodic or white noise. Other bit positions should not be used. **Note** `SOUND` does not 'sanity check' the values passed, so care should be taken to pass in only legal values. **Note** the sound tutorial page offers some additional words which gives more control over the sound chip.
See Also:	None listed

TurboForth

Category 18

Words in category Speech Words in glossary TurboForth Kernal

The following words are listed in this category (3 words found): **Word Types:** S = Standard, I = Immediate , IC = Immediate-compiling.

Word Name	Word Type	Data Stack Signature	Return Stack Signature	Description	Availability V1.0	V1.1	V1.2
SAY	S	address count --	--	Says *count* words from the speech synthesizer's resident vocabulary, from the list at address *address*.	☑	☑	☑
STREAM	S	address count --	--	Streams *count* raw bytes to the speech synthesizer from a buffer in CPU memory beginning at address *address*.	☑	☑	☑
TALKING?	S	-- flag	--	Checks the speech synthesiser and returns `TRUE` if the speech synthesizer is busy, else returns `FALSE`.	☑	☑	☑

SAY

In Speech Words in TurboForth Kernal

Word Name:	SAY
Type:	Standard word
Data Stack Signature:	address count --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Says *count* words from the speech synthesizer's resident vocabulary, from the list at address *address*.

Example:	`: ILF-DATA ( -- addr count )`
	`  DATA 3 $3793 $3F2F $2D19 ;`
	
	`: SPEAK ( -- ) ILF-DATA SAY ;`
	
	`SPEAK`
	
	(says "I like Forth")

Comment:	The address list consists of addresses from the speech synthesizer's resident speech ROM. **Note**: The speech synthesizer FIFO queue is serviced when VDP accessed. Therefore, to ensure smooth access, it is advisable to perform some VDP access periodically. The example given here works okay because the code immediately returns to the command line, where the cursor is flashing on/off, meaning VDP is being written to regularly. **Note**: It is not neccessary to determine if a user's machine has a speech synthesizer fitted or not; if the speech synthesizer is not fitted then all speech related words silently exit with no error.
See Also:	STREAM TALKING?

STREAM

In Speech Words in TurboForth Kernal

Word Name:	STREAM
Type:	Standard word
Data Stack Signature:	address count --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Streams *count* raw bytes to the speech synthesizer from a buffer in CPU memory beginning at address *address*.
Example:	none
Comment:	The byte stream is a previously calculated LPC (Linear Predictive Coding) formatted stream. **Note**: The speech synthesizer FIFO queue is serviced when VDP accessed. Therefore, to ensure smooth access, it is advisable to perform some VDP access periodically. **Note**: It is not neccessary to determine if a user's machine has a speech synthesizer fitted or not; if the speech synthesizer is not fitted then all speech related words silently exit with no error.
See Also:	TERMINAL? SAY

TALKING?

In Speech Words in TurboForth Kernal

Word Name:	TALKING?
Type:	Standard word
Data Stack Signature:	-- flag
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Checks the speech synthesiser and returns TRUE if the speech synthesizer is busy, else returns FALSE.
Example:	none
Comment:	Returns FALSE if the speech synthesizer is not fitted.
See Also:	SAY STREAM

TurboForth

Category 19

Words in category Stack Words in glossary TurboForth Kernal

The following words are listed in this category (16 words found): **Word Types:** S = Standard, I = Immediate , IC = Immediate-compiling.

Word Name	Word Type	Data Stack Signature	Return Stack Signature	Description	Availability V1.0	V1.1	V1.2
.S	S	--	--	Non-destructively displays the contents of the stack.	☑	☑	☑
?DUP	S	a -- a | a a	--	Duplicates the topmost item if it is not 0.	☑	☑	☑
><	S	n -- n	--	Swaps bytes in a cell. E.g. 0x1234 becomes 0x3412.	☑	☑	☑
2DROP	S	a b --	--	Discards the topmost two cells.	☑	☑	☑
2DUP	S	a b -- a b a b	--	Duplicates the topmost two cells.	☑	☑	☑
DEPTH	S	-- depth	--	Places the count of the number of cells on the stack on the stack.	☑	☑	☑
DROP	S	n --	--	Discards the topmost cell.	☑	☑	☑
DUP	S	n -- n	--	Duplicates the topmost cell.	☑	☑	☑
NIP	S	a b -- b	--	Discards the second cell (cell *a*). Equivalent to `SWAP DROP`.	☑	☑	☑
OVER	S	a b -- a b a	--	Copies the second cell to the top. 1 2 becomes 1 2 1.	☑	☑	☑
PICK	S	n – Nth	--	Copies stack cell number n to the top of the stack.	☑	☑	☑
ROLL	S	+n -- n	--	Rolls the stack left +n items. Items rolled out to the left wrap around to the top of the stack.	☑	☑	☑
ROT	S	a b c -- b c a	--	Rotates the top 3 items to the left. The 3rd item rotates into the top position. 1 2 3 becomes 2 3 1.	☑	☑	☑
-ROT	S	a b c -- c a b	--	Rotates the top 3 items to the right. The topmost item rotates into the 3rd position. 1 2 3 becomes 3 1 2.	☑	☑	☑
SWAP	S	a b -- b a	--	Swaps the topmost two cells. 1 2 becomes 2 1	☑	☑	☑
TUCK	S	a b -- b a b	--	Copies the top cell into the 3rd position, moving the stack forward. 1 2 becomes 2 1 2.	☑	☑	☑

.S

In Stack Words in TurboForth Kernal

Word Name:	.S
Type:	Standard word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Non-destructively displays the contents of the stack.
Example:	none
Comment:	none
See Also:	DEPTH

?DUP

In Stack Words in TurboForth Kernal

Word Name:	?DUP
Type:	Standard word
Data Stack Signature:	a -- a | a a
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Duplicates the topmost item if it is not 0.
Example:	none
Comment:	none
See Also:	DUP

><

In Stack Words in TurboForth Kernal

Word Name:	><
Type:	Standard word
Data Stack Signature:	n -- n
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Swaps bytes in a cell. E.g. 0x1234 becomes 0x3412.
Example:	none
Comment:	none
See Also:	><

2DROP

In Stack Words in TurboForth Kernal

Word Name:	2DROP
Type:	Standard word
Data Stack Signature:	a b --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Discards the topmost two cells.
Example:	none
Comment:	none
See Also:	DROP

2DUP

In Stack Words in TurboForth Kernal

Word Name:	2DUP
Type:	Standard word
Data Stack Signature:	a b -- a b a b
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Duplicates the topmost two cells.
Example:	none
Comment:	none
See Also:	2DROP DUP

DEPTH

In Stack Words in TurboForth Kernal

Word Name:	DEPTH
Type:	Standard word
Data Stack Signature:	-- depth
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Places the count of the number of cells on the stack on the stack.
Example:	none
Comment:	For example, if the stack consists of 1 3 5 7 then execution of the word DEPTH results in pushes the value 4 to the stack.
See Also:	.S

DROP

In Stack Words in TurboForth Kernal

Word Name:	DROP
Type:	Standard word
Data Stack Signature:	n --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Discards the topmost cell.
Example:	none
Comment:	none
See Also:	2DROP

DUP

In Stack Words in TurboForth Kernal

Word Name:	DUP
Type:	Standard word
Data Stack Signature:	n -- n
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Duplicates the topmost cell.
Example:	none
Comment:	none
See Also:	2DUP

NIP

In Stack Words in TurboForth Kernal

Word Name:	NIP
Type:	Standard word
Data Stack Signature:	a b -- b
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Discards the second cell (cell *a*). Equivalent to `SWAP DROP`.
Example:	none
Comment:	none
See Also:	None listed

OVER

In Stack Words in TurboForth Kernal

Word Name:	OVER
Type:	Standard word
Data Stack Signature:	a b -- a b a
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Copies the second cell to the top. 1 2 becomes 1 2 1.
Example:	none
Comment:	none
See Also:	None listed

PICK

In Stack Words in TurboForth Kernal

Word Name:	PICK
Type:	Standard word
Data Stack Signature:	n – Nth
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Copies stack cell number n to the top of the stack.
Example:	none
Comment:	`0 PICK` is equivalent to `DUP`. `1 PICK` is equivalent to `OVER`.
See Also:	None listed

ROLL

In Stack Words in TurboForth Kernal

Word Name:	ROLL
Type:	Standard word
Data Stack Signature:	+n -- n
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Rolls the stack left +n items. Items rolled out to the left wrap around to the top of the stack.
Example:	none
Comment:	none
See Also:	None listed

ROT

In Stack Words in TurboForth Kernal

Word Name:	ROT
Type:	Standard word
Data Stack Signature:	a b c -- b c a
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Rotates the top 3 items to the left. The 3rd item rotates into the top position. 1 2 3 becomes 2 3 1.
Example:	none
Comment:	none
See Also:	-ROT

-ROT

In Stack Words in TurboForth Kernal

Word Name:	-ROT
Type:	Standard word
Data Stack Signature:	a b c -- c a b
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Rotates the top 3 items to the right. The topmost item rotates into the 3rd position. 1 2 3 becomes 3 1 2.
Example:	none
Comment:	none
See Also:	ROT

SWAP

In Stack Words in TurboForth Kernal

Word Name:	SWAP
Type:	Standard word
Data Stack Signature:	a b -- b a
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Swaps the topmost two cells. 1 2 becomes 2 1
Example:	none
Comment:	none
See Also:	None listed

TUCK

In Stack Words in TurboForth Kernal

Word Name:	TUCK
Type:	Standard word
Data Stack Signature:	a b -- b a b
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Copies the top cell into the 3rd position, moving the stack forward. 1 2 becomes 2 1 2.
Example:	none
Comment:	none
See Also:	None listed

TurboForth

Category 20

Words in category String Words in glossary TurboForth Kernal

The following words are listed in this category (12 words found): **Word Types:** S = Standard, I = Immediate , IC = Immediate-compiling.

Word Name	Word Type	Data Stack Signature	Return Stack Signature	Description	Availability V1.0	V1.1	V1.2
$.	S	n --	--	Displays the number n as an unsigned 16 bit hexadecimal integer.	☑	☑	☑
.	S	n --	--	Displays n as a signed 16 bit integer in the current number BASE.	☑	☑	☑
."	IC	--	--	Displays the string immediately following it. For use in colon definitions only.	☑	☑	☑
.R	S	n width --	--	Displays the signed number n, right justified by width characters.	☑	☑	☑
ASCII	I	-- ascii	--	Obtains the ASCII code of the first character of the word that follows itself in the input stream.	☑	☑	☑
CHAR	I	-- ascii	--	Returns the ascii value of the first character in the word immediately following.	☑	☑	☑
COUNT	S	addr1 -- addr2 len	--	Converts a counted string at *addr1* to an address/length pair suitable for use with TYPE etc.	☑	☑	☑
N>S	S	n -- addr len	--	Converts a number *n* from the stack into a string equivalent.	☑	☑	☑
S"	IC	-- addr len	--	Stores a string. At run-time, the address *addr* and *length* len of the string are pushed to the stack.	☑	☑	☑
-TRAILING	S	addr len -- addr len1	--	Trims a string of any trailing spaces. Modifies *len* accordingly.	☑	☑	☑
U.	S	n --	--	Displays the value *n* as an unsigned 16 bit integer in the current number BASE.	☑	☑	☑
U.R	S	n width --	--	Displays the unsigned number *n*, right justified by *width* characters.	☑	☑	☑

$.

In String Words in TurboForth Kernal

Word Name:	$.
Type:	Standard word
Data Stack Signature:	n --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Displays the number n as an unsigned 16 bit hexadecimal integer.
Example:	none
Comment:	none
See Also:	.

.

In String Words in TurboForth Kernal

Word Name:	.
Type:	Standard word
Data Stack Signature:	n --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Displays n as a signed 16 bit integer in the current number BASE.
Example:	none
Comment:	none
See Also:	.R U.R U.

."

In String Words in TurboForth Kernal

Word Name:	."
Type:	Immediate compiling word
Data Stack Signature:	--
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Displays the string immediately following it. For use in colon definitions only.
Example:	`: TEST ( -- ) ." Hello, World!" ;`
Comment:	none
See Also:	S"

.R

In String Words in TurboForth Kernal

Word Name:	.R
Type:	Standard word
Data Stack Signature:	n width --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Displays the signed number n, right justified by width characters.

Example:	`: NEAT ( -- )`
	`  CR`
	`  1 3 .R CR`
	`  10 3 .R CR`
	`  100 3 .R CR`
	`;`

Comment:	none
See Also:	.

ASCII

In String Words in TurboForth Kernal

Word Name:	ASCII
Type:	Immediate word
Data Stack Signature:	-- ascii
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Obtains the ASCII code of the first character of the word that follows itself in the input stream.
Example:	`: STAR ( -- 42 ) ASCII * ;`
Comment:	If used on the command line, behaves as per CHAR. If used within a colon definition, compiles the first character of the string immediately following into the colon definition as a literal. The rest of the string is skipped.
See Also:	CHAR

CHAR

In String Words in TurboForth Kernal

Word Name:	CHAR
Type:	Immediate word
Data Stack Signature:	-- ascii
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Returns the ascii value of the first character in the word immediately following.
Example:	`CHAR STAR EMIT` (displays S)
Comment:	Do not use in a colon definition (unless surrounded with [and])
See Also:	ASCII

COUNT

In String Words in TurboForth Kernal

Word Name:	COUNT
Type:	Standard word
Data Stack Signature:	addr1 -- addr2 len
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Converts a counted string at *addr1* to an address/length pair suitable for use with `TYPE` etc.
Example:	none
Comment:	*Addr2* is *addr1*+1, and *len* is the value of the byte found at *addr1*.
See Also:	S"

N>S

In String Words in TurboForth Kernal

Word Name:	N>S
Type:	Standard word
Data Stack Signature:	n -- addr len
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Converts a number *n* from the stack into a string equivalent.

Example:	`99 N>S TYPE`
	(displays 99)
	`32989 N>S TYPE`
	(displays -32547)
	`TRUE UNSIGNED !`
	`32989 N>S TYPE`
	(displays 32989)
	`TRUE ZEROS !`
	`44 N>S TYPE`
	(displays 00044)

Comment:	The address *addr* and length *len* of the string are pushed to the stack. The location of the newly created string should be considered transitory (for example, calling `N>S` again with a different value will overwrite the string). The variable ZEROS is consulted when building the string. If TRUE (or any non-zero value) leading zeros will be included in the string. If the number is negative a leading - sign will be included. In V1.2 It is possible to force numbers to be interpreted as unsigned by setting the `UNSIGNED` variable to true.
See Also:	ZEROS

S"

In String Words in TurboForth Kernal

Word Name:	S"
Type:	Immediate compiling word
Data Stack Signature:	-- addr len
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Stores a string. At run-time, the address *addr* and *length* len of the string are pushed to the stack.

Example:	`: COMPUTER$ ( -- addr len )`
	`  S" TI-99/4A"`
	`;`
	`COMPUTER$ TYPE`
	(displays TI-99/4A)

Comment:	none
See Also:	."

-TRAILING

In String Words in TurboForth Kernal

Word Name:	-TRAILING
Type:	Standard word
Data Stack Signature:	addr len -- addr len1
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Trims a string of any trailing spaces. Modifies *len* accordingly.
Example:	`S" HELLO              " -TRAILING TYPE` (displays the string less the trailing spaces)
Comment:	none
See Also:	Microsoft JET Database Engine error '80040e14' Syntax error (missing operator) in query expression 'ID IN ()'. /lang_ref/view_word.asp, line 123

U.

In String Words in TurboForth Kernal

Word Name:	U.
Type:	Standard word
Data Stack Signature:	n --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Displays the value *n* as an unsigned 16 bit integer in the current number BASE.
Example:	none
Comment:	none
See Also:	BASE .

U.R

In String Words in TurboForth Kernal

Word Name:	U.R
Type:	Standard word
Data Stack Signature:	n width --
Return Stack Signature:	--
Availability:	V1.0 V1.1 V1.2
Description:	Displays the unsigned number *n*, right justified by *width* characters.

Example:	`: NEAT ( -- )`
	`  CR`
	`  1 3 U.R CR`
	`  10 3 U.R CR`
	`  100 3 U.R CR`
	`;`

Comment:	none
See Also:	U.