FlagShip

Object Oriented Database Development System

Cross-Compatible to Unix, Linux and MS-Windows

MULTISOFT

Release 8.1 **Section QRF**

The whole FlagShip 8 manual consist of following sections:

Section	**Content**
GEN	General information: License agreement & warranty, installation and de-installation, registration and support
LNG	FlagShip language: Specification, database, files, language elements, multiuser, multitasking, FlagShip extensions and differences
FSC	Compiler & Tools: Compiling, linking, libraries, make, run-time requirements, debugging, tools and utilities
CMD	Commands and statements: Alphabetical reference of FlagShip commands, declarators and statements
FUN	Standard functions: Alphabetical reference of FlagShip functions
OBJ	Objects and classes: Standard classes for Get, Tbrowse, Error, Application, GUI, as well as other standard classes
RDD	Replaceable Database Drivers
EXT	C-API: FlagShip connection to the C language, Extend C System, Inline C programs, Open C API, Modifying the intermediate C code
FS2	Alphabetical reference of FS2 Toolbox functions
QRF	Quick reference: Overview of commands, functions and environment
PRE	Preprocessor, includes, directives
SYS	System info, porting: System differences to DOS, porting hints, data transfer, terminals and mapping, distributable files
REL	Release notes: Operating system dependent information, predefined terminals
APP	Appendix: Inkey values, control keys, ASCII-ISO table, error codes, dBase and FoxPro notes, forms
IDX	Index of all sections
fsman	The on-line manual "f**sman**" contains all above sections, search function, and additionally last changes and extensions

multisoft Datentechnik, Germany

***Object Oriented Database Development System,
Cross-Compatible to Unix, Linux and MS-Windows***

Section QRF

Manual release: 8.1

For the current program release see your Activation Card,
or check on-line by issuing *FlagShip -version*

Note: the on-line manual is updated more frequently.

Copyright

Trademarks

Headquarter Address

multisoft Datentechnik
Schönaustr. 7
84036 Landshut
Germany

E-mail: support@flagship.de
support@multisoft.de
sales@multisoft.de

Phone: (+49) 0871-3300237

Web: http://www.fship.com

QRF: Quick Reference

Index of FlagShip commands and statements

and their compatibility to previous releases of FlagShip and Clipper (all commands may be abbreviated up to 4 leading characters)

Legend:

FS3	fully compatible with FlagShip 3.x and Clipper'87 + 5.x
+FS3	fully compatible with FlagShip 3.x and Clipper'87, additional extensions with FlagShip 4.3 (and Clipper 5.x)
FS4	new in FlagShip 4.3, fully compatible to Clipper 5.01a and 5.2
++FS4	available only in FlagShip 4.3 (not in Clipper)
*FS4	FlagShip or Unix differences to Clipper
FS5	new for FlagShip5
FS6	new for FlagShip6
FS7	new for FlagShip7
FS8	new for FlagShip8

!	Executes a Unix/Windows command, script or program	FS3
*	Whole line comment	FS3
&&	Inline or whole-line comment	FS3
//	Inline or whole-line comment	FS4
/*..*/	Inline or whole-line (or more lines) comment	FS4
?	Data output (sequential) incl. linefeed	FS3
??	Data output (sequential) w/o linefeed	FS3
?#, ??#, ??##	Sequential output to stderr	FS5
@	Clears a part of the screen	FS3
@...BOX	Draws a custom designed box on the screen	+FS3
@...CLEAR	Clears a screen region	FS3
@...DRAW	Draw lines and graphics in GUI mode	FS7
@...GET	Prepares (and displays) screen oriented data input	+FS3
@...PROMPT	Prepares (and displays) screen oriented menu	+FS3
@...SAY	Displays data on screen (or printer/file)	+FS3
@...TO	Draws boxes on the screen using single/double lines	+FS3
@...GET CHECKBOX	create Checkbox processed by READ	FS5
@...GET COMBOBOX	create Combobox processed by READ	FS5
@...GET LISTBOX	create Listbox processed by READ	FS5
@...GET PUSHBUTTON	create Pushbutton processed by READ	FS5
@...GET RADIOBUTTON	create Radiobutton processed by READ	FS5
@...GET RADIOGROUP	create group of Radiobuttons for READ	FS5
@...GET TBROWSE	create Tbrowse entry for READ	FS5
ACCEPT	Waits for a string to be typed in from the keyboard	FS3
ACCESS METHOD	Declares an access method of a class	++FS4
APPEND BLANK	Adds a new empty record to the end of the current .dbf	FS3
APPEND FROM	Imports records from some other .dbf or ASCII file	+FS3
ANNOUNCE	Declares a module identifier for the linker	FS4

ASSIGN METHOD	Declares an assign method of a class	++FS4
AVERAGE	Calculates an average value of specified .dbf fields	FS3
BEGIN SEQUENCE	Control structure for managing exceptions	+FS3
BREAK	Jumps to the end of a BEGIN SEQUENCE...END structure	+FS3
CALL	Calls a UDF or a C procedure	*FS4
CANCEL	Terminates program execution, closes all files	+FS3
CASE	Selection condition in the DO CASE structure	FS3
CLASS	Declares a user defined class	++FS4
CLEAR ALL	Closes all .dbf's, clears PUBLICs and PRIVATEs	+FS3
CLEAR GETS	Clears the active set of GETs, terminates READ	+FS3
CLEAR MEMORY	Clears all PUBLIC and PRIVATE variables	FS3
CLEAR SCREEN	Clears the screen	FS3
CLEAR TYPEAHEAD	Clears the keyboard buffer	FS3
CLOSE	Closes the specified type of files	+FS3
CLS	Clears the screen	FS4
COMMIT	Writes Unix buffers to the hard disk	+FS3
CONSTANT	Creates protected Public variable	FS5
CONTINUE	Continues the previously started LOCATE search	FS3
COPY FILE	Duplicates any Unix file	FS3
COPY STRUCTURE	Creates an empty .dbf duplicat. the selected .dbf's fields	FS3
COPY STRUCT EXTEND	Creates new structure .dbf containing field info	FS3
COPY TO	Copies parts, or all of the .dbf to a new file	FS3
COUNT	Counts specified records from the actual .dbf	FS3
CREATE	Creates a new structure extended .dbf	FS3
CREATE FROM	Creates an empty .dbf using the structure extended .dbf	FS3
DECLARE	Declares (and initializes) Private arrays	FS3
DELETE	Marks records as "deleted"	FS3
DELETE FILE	Deletes specified Unix file(s) from disk	FS3
DELETE TAG	Deletes specified index tag	++FS4
DIR	Displays a listing of files from the specified path	FS3
DISPLAY	Displays the content of .dbf fields or expressions	FS3
DO	Executes a user defined procedure	FS3
DO CASE	Conditional control structure	FS3
DO WHILE	Looping control structure	+FS3
EJECT	Causes an advance to a new page while printing	FS3
ELSE, ELSEIF	Part of the IF conditional structure	FS3
END[...]	End of the IF, CASE or WHILE structure	FS3
ERASE	Deletes specified Unix file(s) from disk	FS3
EXIT	Terminates a [DO] WHILE / FOR..NEXT loop	FS3
EXIT PROCEDURE	Declare a procedure, executed on program termination	FS4
EXPORT INSTANCE	Declares an export instance of a user defined class	++FS4
EXTERNAL	Declares external procedures to the linker	FS3
FIELD	Declares names of .dbf fields	FS4
FIND	Searches the first key expression in an index file	FS3
FOR..NEXT	Looping control structure	FS3
FUNCTION	Declares a user-defined function	+FS3
GLOBAL...AS	Declares a typed global variable	++FS4

GLOBAL EXTERN...AS	Access to external GLOBAL...AS variables	++FS4
GO, GOTO	Positions the .dbf record pointer to a specific record	FS3
HIDDEN INSTANCE	Declares a hidden instance of a user defined class	++FS4
IF	Conditional control structure	FS3
INIT PROCEDURE	Declare a procedure, executed on program start	FS4
INDEX	Creates an index file that contains .dbf search keys	+FS3
INPUT	Waits for expression to be typed in from the keyboard	FS3
INSTANCE	Declares an instance of a user defined class	++FS4
JOIN	Merges two databases into a new one	FS3
KEYBOARD	Puts a string into the keyboard buffer	+FS3
LABEL FORM	Displays labels defined in a .lbl file	+FS3
LIST	Displays the contents of .dbf fields or expressions	FS3
LOCAL	Declares (and inits) local variables	FS4
LOCAL...AS	Declares typed local variables	++FS4
LOCATE	Sequentially searches the .dbf for the specified condition	FS3
LOOP	Continues the [DO] WHILE / FOR..NEXT loop execution	FS3
MEMVAR	Specifies variables to be Private or Public	FS4
MENU TO	Invokes the prompt-menu system	FS3
METHOD	Declares a method of a class	++FS4
NOTE	Comment line	FS3
ON [ANY] KEY	Assigns a "hot-key" procedure to the specified key	FS5
ON ERROR	simulates FoxPro behavior	FS5
ON ESCAPE	simulates FoxPro behavior	FS5
OTHERWISE	Part of the DO CASE structure	FS3
PACK	Removes all .dbf records marked as "deleted"	FS3
PARAMETERS	Declares Private variables to receive passed parameters	FS3
POP KEY	Restore ON KEY and SET KEY status saved by PUSH KEY	FS5
PRIVATE	Declares (and initializes) Private variables	+FS3
PROCEDURE	Identifies the beginning of a procedure	+FS3
PROTECT INSTANCE	Declares a protect instance of a user defined class	++FS4
PROTECT PUBLIC	Creates protected Public variable	FS5
PROTOTYPE	Prototypes a UDF or a user defined class	++FS4
PUBLIC	Declares (and initializes) Public variables	+FS3
PUSH KEY	Save the ON KEY and SET KEY status	FS5
QUIT	Terminates program execution, closes all files	FS3
READ	Invokes the full screen editing, using the active GET list	+FS3
RECALL	Reinstates the "delete" records	FS3
RECOVER	Exception handling part of the BEGIN SEQUENCE ... END	FS4
REFRESH	Refresh the current screen	FS4
REINDEX	Rebuilds all open indexes in the current working area	FS3
RELEASE	Deletes specified PUBLIC and PRIVATE variables	FS3
RENAME	Renames a Unix file to a new name	FS3
REPLACE	Sets new value in database fields	FS3
REPORT FORM	Displays a formatted report defined in a .frm file	+FS3
RESTORE	Retrieves memory variables from a .mem file	+FS3
RESTORE SCREEN	Displays previously stored screen contents	FS3
REQUEST	Declares external module request to the linker	FS4

RETURN	Terminates a PROCEDURE or FUNCTION	FS3
RUN	Executes a Unix command, script or program	FS3
SAVE	Saves memory variables to a .mem file	+FS3
SAVE SCREEN	Saves the screen content to a SCREEN variable	FS3
SEEK	Searches the first key expression in an index file	FS3
SEEK EVAL	Searches any key expression in index using a code block	++FS4
SELECT	Changes the current working area	FS3
SET ALTERNATE	Redirects output to a text file	+FS3
SET ANSI	Automatic translation of PC8 <-> ANSI character set	FS5
SET AUTOLOCK	Enables/disables the automatic record locking	++FS4
SET BELL	Toggles the sounding of the bell	FS3
SET CENTURY	Toggles the input/display of century digits for dates	FS3
SET CHARSET	Automatic translation of PC8 <-> ANSI input chars	FS5
SET COLOR	Changes the screen color setting	FS3
SET COMMIT	Sets the performance tuning for flushing of changed data	FS8
SET CONFIRM	Toggles confirming the GET input	FS3
SET CONSOLE	Toggles the display of commands to the screen	FS3
SET COORD UNIT	Sets the coordinate unit (row/col, pixel, cm, mm, inch)	FS7
SET CURSOR	Toggles the cursor visibility	FS3
SET DATE	Sets the format for date values	+FS3
SET DBREAD	Automatic translation of PC8 <-> ANSI character set	FS5
SET DBWRITE	Automatic translation of PC8 <-> ANSI character set	FS5
SET DECIMALS	Sets the number of displayed decimal places	FS3
SET DEFAULT	Sets the directory where files are saved and created	FS3
SET DELETED	Toggles the filtering of deleted records	FS3
SET DELIMITERS	Sets/toggles delimiter characters for GET fields	FS3
SET DEVICE	Redirects the output of @..SAY to printer/file	FS3
SET DIRECTORY	Changes the current working directory	++FS4
SET EPOCH	Sets the epoch of date values	FS4
SET EJECT	Performs automatic EJECT on full printer page	FS7
SET ESCAPE	Toggles the state of terminating a READ with the Esc key	FS3
SET EVENTMASK	Specifies which events are considered by Inkey()	FS5
SET EXACT	Toggles the way of comparing character strings	FS3
SET EXCLUSIVE	Enables/disables the multi-user mode of database use	FS3
SET FILTER	Sets a filter condition for database operations	FS3
SET FIXED	Determines how to display numeric values	FS3
SET FONT	Set new GUI output font	FS5
SET FORMAT	Activates a format procedure within READ	FS3
SET FUNCTION	Assigns a string to a function key	FS3
SET GUIALIGN	Align all @..GETs at the same @..SAY column	FS5
SET GUICOLORS	Enable default colors also in GUI mode	FS5
SET GUIPRINT	Toggles direct GUI printer output on/off	FS7
SET GUITRANSL ASCII	automatic ASCII -> ISO conversion	FS5
SET GUITRANSL BOX	draw semi-graphic PC8 @..BOX chars in GUI	FS5
SET GUITRANSL LINES	draw semi-graphic PC8 @..TO chars in GUI	FS5
SET GUITRANSL TEXT	draw semi-graphic PC8 chars in GUI mode	FS5
SET INDEX	Activates one or more indexes for the actual .dbf	FS3

SET INPUT	Enables/disables the keyboard input	FS5
SET INTENSITY	Toggles the enhanced color display for GET and PROMPT	FS3
SET KEY	Assigns a "hot-key" procedure to the specified key	FS3
SET KEYTRANSL	Automatic translation of PC8 <-> ANSI input chars	FS5
SET MARGIN	Sets up the left margin for all printed output	FS3
SET MESSAGE TO	Defines the row and centering for @...PROMPT	FS3
SET MULTILOCKS	Enables/disables multiple record locking	++FS4
SET OPENERROR	Enables/disables RTE on failure of database opening	FS5
SET ORDER	Identifies the main index key	FS3
SET OUTMODE	Designates how to print chars < 32 via ?, qout() etc.	FS5
SET PATH	Sets the path to search when attempting to open files	FS3
SET PIXEL	Enables/disables default coordinates in pixel	FS5
SET PRINTER	Redirects the ?,?? output to printer / file	+FS3
SET PRINT GUI	Toggles direct GUI printer output on/off	FS7
SET PROCEDURE	Directs compiler to compile additional procedure files	FS3
SET RELATION	Relates two working areas using a key expression	FS3
SET RELAT MULT	Relates child databases in 1:N mode	FS7
SET SCRCOMPRESS	Enables/disables compress of SAVE SCREEN images	FS5
SET SCOREBOARD	Toggles messages for READ and MEMOEDIT()	FS3
SET SOFTSEEK	Toggles the soft search criteria for SEEK	FS3
SET SOURCE	Enable/disable automatic ASCII or ANSI translation	FS5
SET TYPEAHEAD	Sets the size of the keyboard buffer	FS3
SET UNIT	Sets the coordinate unit (row/col, pixel, cm, mm, inch)	FS7
SET UNIQUE	Toggles the unique index criteria of index key	FS3
SET WRAP	Defines the wrapping in MENU	FS3
SET ZEROBYTEOUT	Designates how to display \0 character	FS5
SETSTANDARD	Select the "standard" color pair	++FS4
SETENHANCED	Select the "enhanced" color pair	++FS4
SETUNSELECT	Select the "unselected" color pair	++FS4
SKIP	Moves relatively the record pointer in the specified area	FS3
SORT	Sorts records in the current database	FS3
STATIC	Declares (and initializes) STATIC variables	FS4
STATIC..AS	Declares (and initializes) typed STATIC variables	++FS4
STATIC FUNCTION	Identifies a function visible for the .prg file only	FS4
STATIC PROCEDURE	Identifies a procedure visible for the .prg file only	FS4
STORE	Assigns a value to one or more variables	FS3
SUM	Sums a list of numeric expressions of .dbf fields	FS3
TEXT	Displays a block of text	FS3
TOTAL	Sums over specified .dbf records the given expression	FS3
TYPE	Displays the contents of a text file	FS3
UNLOCK	Releases file or record locks	FS3
UPDATE	Updates the current database from another one	FS3
USE	Opens the specified .dbf and associated .dbt, .idx files	+FS3
WAIT	Displays a prompt and waits for a key to be pressed	FS3
WHILE	Looping control structure, equivalent to DO WHILE	+FS3
ZAP	Removes all .dbf records	FS3

User defined commands are supported using the #command and #xcommand preprocessor directives specified per default in the <FlafShip_dir>/include/ std.fh file.

Index of FlagShip standard functions

(functions marked with 4+ may be abbreviated with up to 4 leading chars)

AADD()	Adds a new element at the end of the array	FS4
ABS()	Returns the absolute value of a numeric expression	FS3
ACHOICE()	Executes a pop-up menu	+FS3
ACLONE()	Duplicates an array of any dimension	FS4
ACOPY()	Copies elements from an array to another one	+FS3
ADEL()	Deletes the specified array element	+FS3
ADIR()	Fills an array with info about the specified file	*FS3
AELEMTYPE()	Checks if all array elements are of given type	FS5
AEVAL()	Executes a code block on each array element	FS4
AFIELDS()	Fills the array with info about the .dbf structure	FS3
AFILL()	Fills the specified array with the chosen value	FS3
AINS()	Inserts an undefined element into an array	FS3
ALERT()	Shows a simple dialog box	FS4
ALIAS()	Gets the alias name of the specified working area	FS3
ALLTRIM()	Removes all leading and trailing spaces from a string	FS3
ALTD()	Activates the FlagShip debugger	+FS3
ANSI2OEM()	Convert ANSI string to OEM character set	FS5
ANSITOOEM()	Same as ANSI2OEM()	FS5
APPIOMODE()	Returns the current i/o mode (G/T/B)	FS5
APPMDIMODE()	Determines whether application compiled in MDI mode	FS5
APPOBJECT()	Retrieves the Application object	FS5
ARRAY()	Creates an uninitialized array of specified length	FS4
ASC()	Returns the corresponding ASCII value of a character	FS3
ASCAN()	Seeks a specified value in an array	+FS3
ASIZE()	Resizes an array	FS4
ASORT()	Sorts the specified array's elements in ascending order	+FS3
AT()	Returns the position of a given substring within a string	FS3
ATAIL()	Returns the last element of a given array	FS4
ATANYCHAR()	Search for specified character in string	FS5
AUTOFLOCK()	Perform automatic FLOCK(), user modifiable	++FS4
AUTORLOCK()	Perform automatic RLOCK(), user modifiable	++FS4
AUTOUNLOCK()	Perform automatic UNLOCK and COMMIT, user modifiable	++FS4
BETWEEN()	Checks if expression is within a min/max boundary	FS5
BIN2I()	Converts the string equiv. of 16bit int into numeric var	FS3
BIN2L()	Converts the string equiv. of 32bit long to a numeric var	FS3
BIN2W()	Same as BIN2I() but for unsigned int	FS3
BINAND()	Performs binary AND operation	FS5
BINOR()	Performs binary OR operation	FS5
BINXOR()	Performs binary XOR operation	FS5
BINLSHIFT()	Performs binary left shift	FS5
BINRSHIFT()	Performs binary right shift	FS5
BOF()	Reports attempt to move past the beginning of a .dbf	FS3

BREAK()	Jumps to the END of the BEGIN SEQUENCE structure	+FS3
BROWSE()	Browse database fields in a window	FS4
CDOW()	Finds the name of the day of the week for a date value	FS3
CHR()	Converts an ASCII code to a corresponding character	FS3
CHR2SCREEN()	Converts a string to a screen variable	++FS3
CMONTH() 4+	Finds the name of the month for a date value	FS3
COL()	Reports the current column of the cursor on the screen	FS3
COL2PIXEL()	Converts columns into pixels	FS5
COLOR2RGB()	Transforms color string or object into RGB triplets	FS5
CRC32()	Calculates CRC-32 value of a string or array	FS5
CP437_UTF8()	Converts string from CP437 (ASCII) to UTF-8	FS8
CTOD()	Converts a date string to a data value	FS3
CURDIR()	Returns the current Unix directory	FS3
DATE()	Returns the system date in form of a date value	FS3
DATEVALID()	Test the given date for validity	FS5
DAY()	Extracts the day of the month from a date value	FS3
DBAPPEND()	Appends a new record (see APPEND BLANK)	FS4
DBCLEARFIL()	Clears the active filter (see SET FILTER TO)	FS4
DBCLEARIND()	Deactivates the active indexes (see SET INDEX TO)	FS4
DBCLEARREL()	Deactivates the active relation (see SET RELATION)	FS4
DBCLOSEALL()	Closes all active databases (see CLOSE DATA)	FS4
DBCLOSEAREA()	Closes the active database (see CLOSE, USE)	FS4
DBCOMMIT()	Flushes Unix buffers to disk (see COMMIT)	*FS4
DBCOMMITALL()	Flushes all Unix buffers to disk (see COMMIT)	*FS4
DBCREATE()	Creates a .dbf of given structure described in an array	FS4
DBCREATEIND()	Creates an index file (see INDEX ON)	FS4
DBDELETE()	Marks a .dbf record as "deleted" (see DELETE)	FS4
DBEDIT()	Displays records from one or more database	+FS3
DBEVAL()	Executes a code block for each record of .dbf	FS4
DBF()	Retrieves the true .dbf name of the actual working area	*FS4
DBFILTER()	Retrieves the active filter expression of the selected WA	FS3
DBFINFO()	Returns information about all open databases	FS5
DBFOPEN()	Returns information about all open or selected work areas	FS5
DBGETLOCATE()	Returns the code block of the current Locate condition	++FS4
DBGOBOTTOM()	Moves the .dbf ptr to the last logic.record (GO BOTTOM)	FS4
DBGOTO()	Moves the .dbf pointer to the specified record (see GOTO)	FS4
DBGOTOP()	Moves the .dbf ptr to the first logical record (see GO TOP)	FS4
DBOBJECT()	Returns the object of the current RDD driver	++FS4
DBRECALL()	Removes the "deleted" mark (see RECALL)	FS4
DBREINDEX()	Rebuilds all open indexes in the current .dbf	FS4
DBRELATION()	Retrieves relational expression for the given relation	FS3
DBRELCOUNT()	Retrieves number of relations to childs	FS7
DBRELMULTI()	Retrieves or set 1:n status of already set relation	FS7
DBRLOCKLIST()	Returns an array containing a list of locked records	++FS4
DBRUNLOCK()	Unlock the current or specified record	++FS4
DBSEEK()	Searches the first key expression in an index (see SEEK)	FS4
DBRSELECT()	Retrieves child working area of the specified relation	FS3

DBSELECTAR()	Changes the current working area (see SELECT)	FS4
DBSETDRIVER()	Changes the database driver	FS4
DBSETFILTER()	Sets the filter expression (see SET FILTER..)	FS4
DBSETINDEX()	Opens an index file (see SET INDEX..)	FS4
DBSETLOCATE()	Sets the Locate condition	++FS4
DBSETORDER()	Chooses the main index criteria (see SET ORDER..)	FS4
DBSETRELAT()	Sets a relation to some other .dbf (see SET RELATION)	FS4
DBSKIP()	Moves relatively the .dbf pointer (see SKIP)	FS4
DBSTRUCT()	Creates an array with a structure info of the .dbf	FS4
DBUNLOCK()	Releases file or record locks (see UNLOCK)	FS4
DBUNLOCKALL()	Releases all file or record locks (see UNLOCK ALL)	FS4
DBUSEAREA()	Opens the specified .dbf and associated files (see USE..)	FS4
DEFAULT()	Check and/or set default value of given variable	FS5
DELETED() 4+	Reports if the current record is marked as "deleted"	FS3
DESCEND()	Creates and searches indexes in the reverse order	FS3
DEVOUT()	Outputs a value to the specified device	FS4
DEVOUTPICT()	Outputs a formatted value to the specified device	FS4
DEVPOS()	Moves cursor or printer head to a new position	FS4
DIRECTORY()	Creates an array with the Unix file/directory info	*FS4
DISKSPACE()	Reports the number of free bytes on a specif. file system	FS3
DISPBEGIN()	Marks the beginning of screen buffering	*FS4
DISPBOX()	Draws a box on the screen (see @..TO)	FS4
DISPCOUNT()	Return the number of pending DISPEND() requests	FS4
DISPEND()	Displays the buffered screen output	*FS4
DISPOUT()	Displays an expression on the screen	FS4
DOSERROR()	Reports in number the last operating system error	*FS3
DOW()	Gives ordinal number of the day of the week for a date	FS3
DRAWLINE()	Draw lines in GUI mode	FS5
DTOC()	Converts a date value to a character string	FS3
DTOS()	Converts a date value to a character string "yyyymmdd"	FS3
EMPTY() 4+	Determines if the result of an expression is empty	FS3
EOF()	Finds out if there was an attempt to move past last rec	FS3
ERRORBLOCK()	Creates an error block for an error message	FS4
ERRORBOX()	Display Error-Message in dialog box	FS8
ERRORLEVEL()	Sets the exit error level	FS4
EVAL ()	Executes a code block	FS4
EXECNAME()	Returns the name of the currently executed application	++FS4
EXECPIDNUM()	Returns the process id number of the application	++FS4
EXP()	Evaluates the e^x expression	FS3
FATTRIB()	Retrieves the access rights of an open file	++FS4
FCLOSE()	Closes an open binary file	FS3
FCOUNT() 4+	Determines the number of fields in the current database	FS3
FCREATE()	Creates a new binary file	FS3
FEOF()	Checks for end-of-file after FREAD*()	FS7
FERASE()	Deletes specified Unix file from disk (see ERASE)	FS4
FERROR()	Determines if an error occurred during file operations	FS3
FERROR2STR()	Check for error during file open and return textual descript.	FS8

FIELD() 4+	Returns the field name from the actual .dbf using posit	FS3
FIELDBLOCK()	Creates a SET/GET code block for a field access	FS4
FIELDDECI()	Returns the number of decimal places of specif. field	++FS4
FIELDGET()	Returns a field value from the specified field number	FS4
FIELDGETARR()	Returns an array containing all fields of the curr.record	++FS4
FIELDLEN()	Returns the length of specified field	++FS4
FIELDNAME() 4+	Returns the name of specified field in the current .dbf	FS3
FIELDPOS()	Determines the position of the specified .dbf field	FS4
FIELDPUT()	Assigns a new value to a .dbf field using field position	FS4
FIELDPUTARR()	Replaces all fields of the curr. record	++FS4
FIELDTYPE()	Returns the type of specified field	++FS4
FIELDWBLOCK()	Creates a SET/GET code block for a field write access	FS4
FILE()	Determines whether a file exists in the defined path	*FS3
FILESELECT()	Open file dialog and return selected file(s)	FS8
FINDEXECFILE()	Returns the complete path of an executable	FS5
FKLABEL()	Determines the name of the specified function key	FS4
FKMAX()	Determines the number of available function keys	FS4
FLAGSHIP_DIR()	Returns the path of FlagShip installation directory	FS8
FLOCK() 4+	Locks the actual .dbf before write acc in multi-user mode	FS3
FLOCKF()	Locks/unlocks an open binary file	++FS4
FOPEN()	Opens the specified binary file	FS3
FONTDIALOG()	Modify font properties via user dialog	FS8
FONTNEW()	Creates new font object	FS7
FOUND()	Returns the status of a previous search command	FS3
FREAD()	Reads characters from the binary file into a buffer var	FS3
FREADSTR()	Reads a string from the specified binary file	FS3
FREADTXT()	Reads text lines from the specified ASCII file	++FS4
FRENAME()	Renames a Unix file to a new name (see RENAME)	*FS4
FSEEK()	Moves the binary file pointer to a new position	FS3
FS_SET ("break")	Sets the "break" key	++FS3
FS_SET ("debug")	Sets the "debug" activation key	++FS3
FS_SET ("devel")	Sets/checks the developer/release running mode	++FS3
FS_SET ("escdelay")	Sets/checks the delay time of ESC key	++FS4
FS_SET ("inmap")	Sets/checks the character mapping for keyboard input	++FS3
FS_SET ("intvar")	Sets the returned type of numeric variables	++FS4
FS_SET ("loadl")	Loads a user defined sorting and language table	++FS3
FS_SET ("lower")	Converts file names to lowercase	++FS3
FS_SET ("memcom")	Sets full compatibility of the .mem files to Clipper	++FS4
FS_SET ("outmap")	Sets/checks the character mapping for screen output	++FS3
FS_SET ("pathdeli")	Sets additional SET PATH separators	++FS4
FS_SET ("pathlow")	Converts given path to lowercase	++FS4
FS_SET ("pathupp")	Converts given path to uppercase	++FS4
FS_SET ("print")	Determines the name of the printer spooler file	++FS3
FS_SET ("setlang")	Sets/checks the loaded sorting and language table	++FS3
FS_SET ("shortnam")	Truncates file names to become compatible with DOS	++FS4
FS_SET ("term")	Determines the active terminal or mapping	++FS3
FS_SET ("transl")	Translates the file extension to other one	++FS4

FS_SET ("typeah")	Controls the type-ahead capability for curses	++FS3
FS_SET ("upper")	Converts file names to uppercase	++FS3
FS_SET ("zerobyte")	Enables the usage of chr(0) within a string	++FS4
FWRITE()	Writes the contents of a buffer variable to a binary file	FS3
GETACTIVE()	Determines the actual active GET object (getsys.prg)	FS4
GETALIGN()	Align all @..GETs at the same column	FS5
GETAPPLYKE()	Apply a key to a GET object (getsys.prg)	FS4
GETDOSETKE()	Process SET KEY during GET editing (getsys.prg)	FS4
GETENV() 4+	Returns the contents of a Unix environment variable	*FS3
GETENVARR()	Returns the contents of all environment variables	FS5
GETFUNCTION()	Returns the string set by SET FUNCTION	FS5
GETPOSTVAL()	Check a post-valid condition (getsys.prg)	FS4
GETPREVALI()	Check a pre-valid condition (getsys.prg)	FS4
GETREADER()	User defined READ for one GET (see getsys.prg)	FS4
GUIDRAWLINE()	Draw lines in GUI mode	FS5
HARDCR()	Replaces all soft CR within a string with hard CR	FS3
HEADER()	Retrieves the size of the header of a database file	FS3
HEX2NUM()	Converts a string of hex values to numeric equivalence	FS5
I2BIN()	Transforms an integer to a character string	FS3
IF() , IIF()	Returns either of the expressions depending on condition	FS3
INDEXCHECK()	Check the consistence of database and its indices	++FS4
INDEXCOUNT()	Determines the number of used indices in the current WA	++FS4
INDEXDBF()	Determines the database name of an index file	++FS4
INDEXEXT()	Determines the type of indexes used in an application	*FS3
INDEXKEY()	Returns the key expression for a specified index	FS3
INDEXNAMES()	Determines all used indexes in the current working area	++FS4
INDEXORD()	Determines the ordinal number of the main index	FS3
INFOBOX()	Display infobox dialog, similar to Alert()	FS5
INKEY()	Reads a character from the keyboard buffer	FS3
INKEY2STR()	Translates inkey number to human readable string	FS5
INKEY2READ()	Re-define/set own text for Inkey2str()	FS5
INKEYTRAP()	same as Inkey() but process SET KEY trap	FS5
INSTDCHAR()	Read one character (with wait) from stdin	FS5
INSTDSTRING()	Read a string (with a wait for ENTER) from stdin	FS5
INT()	Converts a real value to integer	FS3
ISALPHA()	Checks if the specified string begins with an alpha char	FS3
ISBEGSEQ()	Checks if a BREAK can be executed	++FS4
ISCOLOR()	Determines if the terminal definition has color capab.	FS3
ISDBEXCL()	Determines if the database is open in exclusive mode	++FS4
ISDBFLOCK()	Determines if the database is locked by FLOCK()	++FS4
ISDBRLOCK()	Determines if the database record is locked by RLOCK()	++FS4
ISDIGIT()	Checks if the specified string begins with a digit	FS4
ISFUNCTION()	Determines if the specified UDF is available (linked to)	++FS4
ISGUIMODE()	Checks if the application is running in GUI mode	FS5
ISLOWER()	Checks if the specified string begins with a lowercase	FS3
ISOBJCLASS()	Determines the class name of an object variable	++FS4
ISOBJEQUIV()	Checks if two objects are equivalent	FS5

ISOBJPROPER()	Determines the available properties of a class	++FS4
ISPRINTER()	Determines if printer is ready (FS: always true, spool)	*FS3
ISUPPER()	Checks if the specified string begins with an uppercase	FS3
L2BIN()	Transforms an integer to a 4 bytes string	FS3
LASTKEY()	Returns the code of the last depressed key	FS3
LASTREC() 4+	Retrieves the number of phys. records in the current .dbf	FS3
LEFT()	Extracts specified number of leading chars from string	FS3
LEN()	Retrieves the length of a string or size of an array	FS3
LISTBOX()	Instantiates Listbox or Combobox class	FS5
LOCK()	Locks actual record before writing it in multiuser mode	FS3
LOG()	Evaluates the argument and returns its natural logarithm	FS3
LOWER() 4+	Converts a string to lowercase	FS3
LTRIM() 4+	Removes all leading spaces from a string	FS3
LUPDATE()	Retrieves the last modification date of the current .dbf	FS3
MACROEVAL()	Evaluates macro, similar to &(var)	FS5
MACROSUBST()	Substitute macro, similar to "...&macro..." in text	FS5
MAX()	Determines the greater of two numbers or date values	FS3
MAXCOL()	Determines the last available screen column	FS4
MAX_COL()	Determines the last available screen column	++FS3
MAXROW()	Determines the last available screen line	FS4
MAX_ROW()	Determines the last available screen line	++FS3
MCOL()	Determines the mouse column position	FS5
MDBLCK()	Determine the double-click speed threshold of the mouse	FS5
MDICLOSE()	Close current or specified MDI sub-window	FS5
MDIOPEN()	Open new MDI sub-window	FS5
MDISELECT()	Select/set focus to a MDI sub-window	FS5
MEMOEDIT()	Displays or edits strings or memo fields	+FS3
MEMOLINE()	Extracts a formatted line from string or memo field	FS3
MEMOREAD()	Reads a text file from the disk to a character variable	FS3
MEMORY()	Determines free and used memory	FS7
MEMODECODE()	Decodes the string previously coded by MemoEncode()	FS5
MEMOENCODE()	Encodes string so that it is free of chr(0) and chr(27)	FS5
MEMOTRAN()	Replaces all carriage return/line feed pairs	FS3
MEMOWRIT()	Writes a string or memo field to a specified text file	FS3
MEMVARBLOCK()	Returns a SET/GET code block for a memory variable	FS4
MHIDE()	Hide the mouse pointer/cursor	FS5
MIN()	Determines the lower of two numbers or date values	FS3
MINMAX()	Checks if expression is within a min/max boundary	FS5
MLCOUNT()	Counts the number of lines in a string or a memo field	FS3
MLCTOPOS()	Determines position of a substring in a format. string	FS4
MLEFTDOWN()	Determine the status of the left mouse button	FS5
MLPOS()	Determines the line position within a formatted string	FS3
MOD()	Returns the dBASE III modulo of two numbers	FS3
MONTH() 4+	Extracts the month of the year from a date value	FS3
MPOSTOLC()	Determines the col/row position within a formatted string	FS4
MPRESENT()	Determine if a mouse is present	FS5
MRESTSTATE()	Re-establish the previous state of a mouse	FS5

MRIGHTDOWN()	Determine the status of the right mouse button	FS5
MROW()	Determines the mouse row position	FS5
MSAVESTATE()	Save the current state of a mouse	FS5
MSETCURSOR()	Determine or set mouse cursor visibility and/or shape	FS5
MSETPOS()	Set a new position for the mouse cursor	FS5
MSHOW()	Display the mouse pointer	FS5
MSTATE()	Return the current mouse state	FS5
NETERR()	Checks the error status in multi-user environment	+FS3
NETNAME()	Retrieves the current user & workstation identification	*FS3
NEXTKEY()	Looks up the next key in the keyboard buffer	FS3
NOSNOW()	DOS: snow on color terminal, FS: see scrnmode.prg	*FS4
NUM2INT()	Converts floating point variable to IntVar	++FS4
NUM2HEX()	Converts number to hex string	FS5
OEM2ANSI()	Convert OEM/PC8 string to ISO/ANSI character set	FS5
ONKEY()	Redirect key to UDF, similar to SET KEY/ON KEY commands	FS5
ORDBAGEXT()	Return the default order bag RDD extension	FS5
ORDBAGNAME()	Return the order bag name of a specific order	FS5
ORDCONDSET()	Set the condition and scope for an order	FS5
ORDCREATE()	Create an order in an order bag	FS5
ORDDESTROY()	Remove a specified order from an order bag	FS5
ORDDESCEND()	Return or change the descending flag of an order	FS5
ORDFOR()	Return the FOR expression of an order	FS5
ORDISUNIQUE()	Return the status of the unique flag for a given order	FS5
ORDKEY()	Return the key expression of an order	FS5
ORDKEYADD()	Add a key to a custom built order	FS5
ORDKEYCOUNT()	Return the number of keys in an order	FS5
ORDKEYDEL()	Delete a key from a custom built order	FS5
ORDKEYGOTO()	Move to a record specified by its logical record number	FS5
ORDKEYNO()	Get the logical record number of the current record	FS5
ORDKEYVAL()	Get key value of the current record from curr. order	FS5
ORDLISTADD()	Add orders to the order list	FS5
ORDLISTCLEAR()	Clear the current order list	FS5
ORDLISTREBUI()	Rebuild all orders in the list of the current work area	FS5
ORDNAME()	Return the name of an order in the order list	FS5
ORDNUMBER()	Return the position of an order in the curr. order list	FS5
ORDSCOPE()	Set or clear the boundaries for scoping key values	FS5
ORDSETFOCU()	Set focus to an order in an order list	FS5
ORDSETRELAT()	Relate a specified work	FS5
ORDSKIPUNIQUE()	Move record pointer to the next or previous unique key	FS5
OS()	Returns the name of the operating system	*FS3
OUTERR()	Outputs to stderr (as the ?? command, re-routable)	*FS4
OUTSTD()	Outputs to stdout (as the ?? command, w/o SET PRINT..)	*FS4
PADC()	Fills the beginning and the end of a string with chars	FS4
PADL()	Fills the beginning of a string with characters	FS4
PADR()	Fills the end of string with characters	FS4
PARAM()	Retrieve the value of specific parameter number	FS5
PARAMETERS()	Retrieves the number of parameters passed to a procedure	FS3

PCALLS()	Determines the call stack	++FS4
PCOL()	Reports the current printer column position	FS3
PCOUNT() 4+	Retrieves the number of parameters passed to a procedure	FS3
PIXEL2COL()	Calculates given pixel value to columns	FS5
PIXEL2ROW()	Calculates given pixel value to rows	FS5
PRINTGUI()	Toggles direct GUI output on/off, output to GDI printer	FS7
PRINTSTATUS()	Determines whether printer is available	FS5
PROCLINE()	Retrieves the current (or calling) program line number	+FS3
PROCNAME()	Retrieves the current (or calling) program name	+FS3
PROCFILE()	Retrieves the current (or calling) source file name	++FS4
PROCSTACK()	Retrieves the current (or calling) callstack as string	FS5
PROPER()	Set the first char of all words in string to upper case	FS5
PROW()	Reports the current printer row position	FS3
PUSHBUTTON()	Creates PushButton	FS8
QOUT()	Does sequential output, see ?	FS4
QQOUT()	Does sequential output, see ??	FS4
READSAVE()	very similar to READ SAVE command	FS5
READUPDATED()	Determines whether the READ fields were changed	FS5
RAT()	Returns the last position of a substring within a string	FS3
RANGECHECK()	Performs RANGE checking within READ (see getsys.prg)	FS4
RDDLIST()	Determines the list of currently used RDD drivers	++FS4
RDDSETDEFA()	Sets the default RDD driver (user modifiable)	++FS4
READEXIT()	Toggles the cursor keys as exit from a READ (getsys.prg)	+FS3
READINSERT()	Toggles the current insert mode setting (getsys)	+FS3
READKEY()	The dBASE III equivalent to LASTKEY() (see getsys.prg)	FS4
READMODAL()	Performs a user defined READ for one GET (getsys.prg)	FS4
READSELECT()	Select specified GET element during READ	FS7
READVAR()	Retrieves the name of a GET or MENU variable (getsys)	+FS3
RECCOUNT() 4+	Retrieves the number of phys. records in the current .dbf	FS3
RECNO() 4+	Retrieves the physical record number in the current .dbf	FS3
RECSIZE()	Retrieves the record size of the current database file	FS3
REPLICATE() 4+	Forms a new string by repeating a given string n-times	FS3
RESTSCREEN()	Restores a screen region from memory variable	*FS3
RIGHT()	Extracts the rightmost number of characters from string	FS3
RLOCK() 4+	Locks actual record before writing it in multiuser mode	FS3
ROUND() 4+	Rounds a numeric value to the specif. # of decimal places	FS3
ROW()	Reports the current row of the cursor on the screen	FS3
ROW2PIXEL()	Converts rows into pixels	FS5
RTRIM() 4+	Removes all trailing spaces from a string	FS3
SAVESCREEN()	Saves a specified screen region to a variable	*FS3
SCRDOS2UNIX()	Converts SaveScreen() content from DOS to Unix	++FS4
SCREEN2CHR()	Converts a screen variable to a character string	++FS3
SCRUNIX2DOS()	Converts SaveScreen() content from Unix to DOS	++FS4
SCROLL()	Scrolls a specified screen region up, down, or clears it	+FS3
SECONDS() 4+	Reports how many seconds elapsed since midnight	*FS3
SECONDSCPU()	Reports how many CPU seconds elapsed since prog start	++FS3
SELECT() 4+	Gets the working area number for the specified alias	FS3

SET()	Reports/sets global default settings	FS4
SETANSI()	Controls how to read from or write to database	FS5
SETBLINK()	Sets the interpretation of * color, see scrnmode.prg	*FS4
SETCANCEL()	Toggles program termination with Ctrl-K on or off	FS3
SETCOLOR()	Reports/sets the current color setting	FS3
SETCOLORBA()	Report/redefine the GUI background	FS5
SETCOL2GET()	Reports/sets the color setting for GET (unsel/enhanc)	++FS4
SETCURSOR()	Reports/sets the cursor mode	*FS4
SETEVENT()	Changes the behavior in handling events	FS5
SETFONT()	Returns current and/or set new font (see SET FONT)	FS8
SETKEY()	Assigns an action code block to a hot-key	FS4
SETKEYREST()	Restore SetKey's saved by SETKEYSAVE()	FS8
SETKEYSAVE()	Save SET KEYs and ON KEYs for later restore	FS8
SETMODE()	Changes the screen mode (see scrnmode.prg)	*FS4
SETPOS()	Moves the cursor to a new position	FS4
SETPRC() 4+	Sets the printer row and column to the specified value	FS3
SETVAREMPTY()	Sets a variable to empty status	FS5
SLEEP()	Suspend application for given seconds	FS5
SLEEPMS()	Similar to SLEEP() but accepts values in Milliseconds	FS5
SOUNDEX()	Converts character strings to a soundex code	FS3
SPACE() 4+	Forms a string consisting of the given number of spaces	FS3
SQRT()	Evaluates the argument and returns its square root	FS3
STATBARMSG()	Display message in status bar	FS5
STATUSMESSAGE()	Display message in status bar or MENU TO line	FS5
STR()	Converts a numeric expression to a character string	FS3
STRLEN()	Retrieves the length of a string, same as LEN()	FS5
STRLEN2COL()	Retrieves the true length of a string in columns	FS5
STRLEN2PIX()	Retrieves the true length of a string in pixels	FS5
STRLEN2SPACE()	Retrieves the number of spaces required to fill string	FS5
STRPEEK()	Determines a single character in a string	++FS4
STRPOKE()	Replaces a string character with other one	++FS4
STRTRAN()	Searches and replaces within a character string	FS3
STRZERO()	Converts a numeric value to a string with leading zeros	FS3
STUFF()	Performs delete, insert and replace within a string	FS3
SUBSTR() 4+	Extracts the specified part of the given string	FS3
TBROWSEARR()	Instantiates Tbrowse object for array processing	FS5
TBROWSEDB()	Instantiates Tbrowse object for database processing	FS3
TBROWSENEW()	Instantiates Tbrowse object for general purpose	FS3
TEMPFILENAME()	Determines an unique file name	++FS4
TIME()	Reports the system time	FS3
TONE()	Produces a beep	*FS3
TRANSFORM() 4+	Formats an expression according to the given PICTURE	FS3
TRIM()	Removes all trailing spaces from a string	FS3
TRUEPATH()	Returns converted path, file and drive substitution	++FS4
TYPE()	Retrieves the type of macro evaluated expression	FS3
UPDATED()	Reports changes within READ (see getsys.prg)	+FS3
UPPER() 4+	Converts a string to uppercase	FS3

USED()	Determines if a .dbf is open in the selected working area	FS3
USERSACTIV()	Determines the number of actually active users	++FS4
USERSDBF()	Determines the number of users using a .dbf	++FS4
USERSMAX()	Reports the max.no of allowable users (License restrict.)	++FS4
VAL()	Converts a string to a numeric value	FS3
VALTYPE()	Retrieves the type of any FlagShip variable	FS4
VERSION()	Determines the actual FlagShip version	*FS4
WORD()	Converts a numeric value to int	*FS4
YEAR()	Extracts the year from a date value	FS3
_DISPLARR()	Formats an array for displaying	FS5
_DISPLARRERR()	Formats an array for displaying and print it on stderr	FS5
_DISPLARRSTD()	Formats an array for displaying and print it on screen	FS5
_DISPLOBJ()	Formats an object for displaying	FS5
_DISPLOBJERR()	Formats an object for displaying and print it on stderr	FS5
_DISPLOBJSTD()	Formats an object for displaying and print it on screen	FS5
FSGET()	Prepares the @...GET objects	++FS4
__ACCEPT()	Same functionality as ACCEPT	FS4
__DBPACK()	Same functionality as PACK	FS4
__DBZAP()	Same functionality as ZAP	FS4
__EJECT()	Same functionality as EJECT	FS4
__KEYBOARD()	Same functionality as CLEAR TYPEAHEAD/KEYBOARD	FS4
__KILLREAD()	see std.fh, CLEAR GETS and getsys.prg	FS4
__MCLEAR()	Same functionality as CLEAR MEMORY	FS4
__MENUTO()	Same functionality as MENU TO	FS4
__QUIT()	Same functionality as QUIT	FS4
__SETCENTURY()	Same functionality as SET CENTURY	FS4
__SETFORMAT()	Same functionality as SET FORMAT (getsys)	FS4
__SETFUNCTION()	Same functionality as SET FUNCTION	FS4
__WAIT()	Same functionality as WAIT	FS4
__XRESTSCREEN()	Same functionality as RESTORE SCREEN	FS4
__XSAVESCREEN()	Same functionality as SAVE SCREEN	FS4

Properties of the GET class

GETNEW()	Creates a new GET object	FS4
oGet := GET {..}	Instantiate new GET object, equivalent to GETNEW()	++FS4
get:BADDATE	Reports an invalid date in the edit buffer	FS4
get:BLOCK	Code block connecting the GET to the variable	FS4
get:BUFFER	Character oriented edit buffer	FS4
get:CARGO	User defined values	FS4
get:CHANGED	Reports changes within the edit buffer	FS4
get:COL	Screen column of the Get object	FS4
get:COLORSPEC	Color specification of the Get object	FS4
get:DECPOS	Position of the decimal point (1..n, 0)	FS4
get:EXITSTATE	Reports the exit status of edit	FS4
get:HASFOCUS	Reports the edit activity of the object	FS4
get:MINUS	Reports that a minus sign was typed in	FS4
get:NAME	Reports the name of the actual Get variable	FS4
get:ORIGINAL	Reports the original value of the Get variable	FS4
get:PICTURE	Format picture string	FS4
get:POS	Reports the actual cursor position in the buffer (0..n)	FS4
get:POSTBLOCK	Code block for the VALID checking	FS4
get:PREBLOCK	Code block for the WHEN condition	FS4
get:REJECTED	Reports losing the last character in buffer	FS4
get:ROW	Screen row of the Get object	FS4
get:SUBSCRIPT	Info about array GET names	FS4
get:READER	Code block for editing the Get object	FS4
get:TYPE	Reports the type of the Get variable	FS4
get:TYPEOUT	Reports an invalid cursor position	FS4
get:ASSIGN()	Assigns a new value to the Get variable	FS4
get:BACKSPACE()	Moves the cursor one space back and deletes char	FS4
get:BUFFER()	Returns the actual buffer contents	FS4
get:COLORDISP()	Changes the color specification of the Get object	FS4
get:DELETE()	Deletes the character on cursor position	FS4
get:DELWORDRIG()	Deletes one word right	FS4
get:DISPLAY()	Re-displays the actual Get object	FS4
get:END()	Moves cursor to the end of the Get field	FS4
get:HOME()	Moves cursor to the beginning of the Get field	FS4
get:INSERT()	Inserts one character into the edit buffer	FS4
get:KILLFOCUS()	Deactivates the edit mode of Get object	FS4
get:LEFT()	Moves cursor left	FS4
get:OVERSTRIKE()	Overstrikes the actual character in edit buffer	FS4
get:RESET()	Resets all status information	FS4
get:RIGHT()	Moves cursor right	FS4
get:SETFOCUS()	Activates the edit mode of the Get object	FS4
get:TODECPOS()	Moves cursor past decimal point	FS4

get:UNDO()	Resets buffer to the original value	FS4
get:UPDATEBUFF()	Resets buffer to the actual value of the Get variable	FS4
get:UNTRANSFOR()	Converts buffer to the date value of the Get variable	FS4
get:VARGET()	Returns the contents of the actual Get variable	FS4
get:VARPUT()	Stores a new value into the Get variable	FS4
get:WORDLEFT()	Moves the cursor one word left	FS4
get:WORDRIGHT()	Moves the cursor one word right	FS4

Properties of the TBROWSE class

TBROWSENEW()	Creates a new empty TBrowse object	FS4
TBROWSEARR()	Creates a TBrowse object for data in array	++FS5
TBROWSEDB()	Creates a TBrowse object for source data of .dbf	FS4
oTb := TBROWSE{..}	Instantiate a TBrowse object, equiv.to TBROWSENEW()	++FS4
tb:AUTOLITE	Logical value controlling the light-bar	FS4
tb:CARGO	User defined values	FS4
tb:COLCOUNT	Number of columns	FS4
tb:COLORSPEC	String containing color attributes	FS4
tb:COLPOS	Position of the actual column	FS4
tb:COLSEP	String for the column separator	FS4
tb:FOOTSEP	String for the bottom line separator	FS4
tb:FREEZE	Position of the freezed column	FS4
tb:GOBOTTOMBL	Code block for moving to the end	FS4
tb:GOTOPBLOCK	Code block for moving to the top	FS4
tb:HEADSEP	String for the header separator	FS4
tb:HITBOTTOM	Reports if the end was reached	FS4
tb:HITTOP	Reports if the top was reached	FS4
tb:LEFTVISIB	Leftmost visible, variable column	FS4
tb:NBOTTOM	Last screen row of the TBrowse window	FS4
tb:NLEFT	Left screen column of the TBrowse window	FS4
tb:NRIGHT	Right screen column of the TBrowse window	FS4
tb:NTOP	First screen row of the TBrowse window	FS4
tb:RIGHTVISIB	Rightmost visible, variable column	FS4
tb:ROWCOUNT	Number of rows to display data	FS4
tb:ROWPOS	Position of the actual data row	FS4
tb:SKIPBLOCK	Code block for skipping trough the data	FS4
tb:STABLE	Reports the stable state of TBrowse	FS4
tb:ADDCOLUMN()	Inserts a new TBColumn object	FS4
tb:COLORRECT()	Changes the color for a visible field	FS4
tb:COLWIDTH()	Returns the width of the column	FS4
tb:CONFIGURE()	Resets the internal state	FS4
tb:DEHILITE()	Disables the light-bar	FS4
tb:DELCOLUMN()	Deletes one data column	FS4
tb:DOWN()	Moves the cursor one line down	FS4
tb:END()	Moves the cursor to the rightmost visible column	FS4
tb:FORCESTABL()	Forces the stabilization	FS4
tb:GETCOLUMN()	Returns the TBColumn object	FS4
tb:GOBOTTOM()	Moves the pointer to the end of data position	FS4
tb:GOTOP()	Moves the pointer to the first data position	FS4
tb:HILITE()	Enables the light-bar	FS4
tb:HOME()	Moves the cursor to the leftmost visible column	FS4
tb:INSCOLUMN()	Inserts a new column	FS4

tb:INVALIDATE()	Rebuilds the TBrowse window from the internal buffer	FS4
tb:LEFT()	Moves the cursor one column left	FS4
tb:PAGEDOWN()	Moves the cursor to the next page	FS4
tb:PAGEUP()	Moves the cursor to the previous page	FS4
tb:PANEND()	Moves the cursor to the last available column	FS4
tb:PANHOME()	Moves the cursor to the first available column	FS4
tb:PANLEFT()	Scrolls the TBrowse window one column left	FS4
tb:PANRIGHT()	Scrolls the TBrowse window one column right	FS4
tb:REFRESHALL()	Refreshes all data on the next stabilize()	FS4
tb:REFRESHCURR()	Refreshes act.line contents on the next stabilize()	FS4
tb:RIGHT()	Moves the cursor one column right	FS4
tb:SETCOLUMN()	Replaces a data column by a new one	FS4
tb:STABILIZE()	Executes partial stabilizing	FS4
tb:UP()	Moves the cursor one line up	FS4

Properties of the TBCOLUMN class

TBCOLUMNNEW()	Creates a new column object	FS4
oTb := TBCOLUMN{..}	Instantiate TBColumn object, equiv.to TBCOLUMNNEW()	++FS4
tc:BLOCK	Code block for receiving the data	FS4
tc:CARGO	User defined values	FS4
tc:COLORBLOCK	Code block for receiving the color definitions	FS4
tc:COLSEP	String for separating columns	FS4
tc:DEFCOLOR	An array of color attributes	FS4
tc:FOOTING	String for separating the footing line	FS4
tc:FOOTSEP	String for separating footing columns	FS4
tc:HEADING	String for separating the header line	FS4
tc:HEADSEP	String for separating header columns	FS4
tc:WIDTH	Width of the column	FS4

Properties of the ERROR class

ERRORNEW()	Creates a new Error object	FS4
oEr := ERROR{..}	Instantiate new Error object, equiv.to ERRORNEW()	++FS4
err:ARGS	Array containing the function's or operation's parameters	FS4
err:CANDEFAULT	Reports whether able to recover	FS4
err:CANRETRY	Reports whether able to retry	FS4
err:CANSUBSTIT	Reports the replacement possibility by a default value	FS4
err:CARGO	User defined values	FS4
err:DESCRIPTION	Contains the error message	FS4
err:FILENAME	Contains the filename which caused the error	FS4
err:GENCODE	Contains the internal error number	*FS4
err:OPERATION	Contains the description of the failed operation	FS4
err:OSCODE	Contains the operating system error code, if available	FS4
err:SEVERITY	Contains the error severity	FS4
err:SUBCODE	Contains the error number from the subsystem	FS4
err:SUBSYSTEM	Contains the name of the error subsystem	FS4
err:TRIES	Contains the allowable number of retries	FS4

Properties of DataServer and DBserver class

DBSERVERNEW()	Creates a new DbServer object	++FS4
DBFIDXNEW()	Creates a new DbfIdx object	++FS4
oRdd := DBDIDX{..}	Instantiate DbfIdx object, equiv.to DBFIDXNEW()	++FS4
oRdd:ALIAS	Access/assign the alias if the work area	++FS4
oRdd:APPEND()	Appends an empty record	++FS4
oRdd:APPENDDB()	Appends data from other .dbf	++FS4
oRdd:APPENDDELIMITED()	Appends data from ascii file	++FS4
oRdd:APPENDSDF()	Appends data from ascii file	++FS4
oRdd:ASSTRING()	Similar to oRdd:NAME	++FS4
oRdd:AVERAGE()	Calculates an average	++FS4
oRdd:AXIT()	Performs garbage collection before destroying object	++FS4
oRdd:BOF	Determines if the BOF flag is set	++FS4
oRdd:CARGO	Carries user defined data	++FS4
oRdd:CLEARFILTER()	Clears the filter condition	++FS4
oRdd:CLEARINDEX()	Frees the associated indices	++FS4
oRdd:CLEARLOCATE()	Clears the Locate condition	++FS4
oRdd:CLEARRELATION()	Clears the Relations	++FS4
oRdd:CLEARSCOPE()	Clears the global scopes	++FS4
oRdd:CLOSE()	Closes the databases	++FS4
oRdd:COMMIT()	Flushes data to disk	++FS4
oRdd:CONCURRENCYCONTROL	Controls the lock mode	++FS4
oRdd:CONTINUE()	Continue the pending Locate	++FS4
oRdd:COPYDB()	Copies records to other database	++FS4
oRdd:COPYDELIMITED()	Copies/exports records to ascii file	++FS4
oRdd:COPYSDF()	Copies/exports records to ascii file	++FS4
oRdd:COPYSTRUCTURE()	Creates an empty database	++FS4
oRdd:COUNT()	Counts records which fulfill specif.condition	++FS4
oRdd:CREATEDB()	Creates an empty database	++FS4
oRdd:CREATEINDEX()	Creates an index file	++FS4
oRdd:CREATEORDER()	Creates an index file with multiple orders	++FS4
oRdd:DBSTRUCT()	Returns an array containing the database structure	++FS4
oRdd:DELETE()	Deletes the current / a range of records	++FS4
oRdd:DELETEALL()	Marks all records as deleted	++FS4
oRdd:DELETED	Determines if the curr record is marked as deleted	++FS4
oRdd:DELETEORDER()	Destroys a multiple-order index file	++FS4
oRdd:DRIVER	Retrieves the name of currently used RDD	++FS4
oRdd:EOF	Determines if the BOF flag is set	++FS4
oRdd:ERRINFO	Returns the error object of previous error	++FS4

oRdd:ERROR()	Error handling method	++FS4
oRdd:EVAL()	Evaluates a code block for each record	++FS4
oRdd:FCOUNT	Determines the number of fields per record	++FS4
oRdd:FIELDGET()	Retrieves the value of specified field	++FS4
oRdd:FIELDINFO()	Retrieves information about a field	++FS4
oRdd:FIELNAME()	Retrieves the name of specified field	++FS4
oRdd:FIELPOS()	Retrieves the ordinal position of specified field	++FS4
oRdd:FIELDPUT()	Assigns a given value to a field	++FS4
oRdd:FILTER	Sets/retrieves the filter condition string	++FS4
oRdd:FLOCK()	Locks all records of the database	++FS4
oRdd:FORBLOCK	Sets/retrieves the general FOR block	++FS4
oRdd:FOUND	Determines the success of previous Seek, Locate	++FS4
oRdd:GETARRAY()	Assigns the values of an array to all record fields	++FS4
oRdd:GETARRFIELDS()	Assigns values of all fields in curr.record to arr	++FS4
oRdd:GETLOCATE()	Retrieves the code block of Locate	++FS4
oRdd:GETLOOKUPTABLE()	Assigns values of several record to an array	++FS4
oRdd:GOBOTTOM()	Moves the database pointer to the last logical record	++FS4
oRdd:GOTO()	Moves the database pointer to specified record	++FS4
oRdd:GOTOP()	Moves the database pointer to the first logical record	++FS4
oRdd:HEADER	Determines the header size	++FS4
oRdd:INDEXCHECK()	Checks the index integrity	++FS4
oRdd:INDEXCOUNT	Determines the number of open indices	++FS4
oRdd:INDEXEXT	Determines the default index extension	++FS4
oRdd:INDEXKEY	Determines the key expression	++FS4
oRdd:INDEXKEY()	Determines the key expression	++FS4
oRdd:INDEXORD()	Returns the ordinal position of controlling index	++FS4
oRdd:INFO()	Returns a general information about the data server	++FS4
oRdd:INIT()	Initializes the object, opens the database	++FS4
oRdd:ISRELATION	Determines, if a relation is active	++FS4
oRdd:JOIN()	Creates a new database by merging two databases	++FS4
oRdd:LASTREC	Determines the number of records in the curr. database	++FS4
oRdd:LOCATE()	Searches for the first record meeting specif. condit.	++FS4
oRdd:LOCKCURRENTRECORD()	Locks the current record	++FS4
oRdd:LUPDATE	Retrieves the last modification date	++FS4
oRdd:NAME	Returns the main part of the database name	++FS4
oRdd:NOIVARGET()	Provides general error interception	++FS4
oRdd:NOIVARPUT()	Provides general error interception	++FS4
oRdd:NOMETHOD()	Provides general error interception	++FS4
oRdd:ORDERBOTTOMSCOPE	Controls the index visibility	++FS4

oRdd:ORDERDESCEND()	Controls the descended sorting	++FS4
oRdd:ORDERINFO()	Returns an info about orders and index files	++FS4
oRdd:ORDERISUNIQUE()	Determines the Unique status	++FS4
oRdd:ORDERKEYCOUNT()	Returns the number of keys in an order	++FS4
oRdd:ORDERKEYGOTO()	Moves to specified logical record	++FS4
oRdd:ORDERKEYNO()	Returns the logical rec.number of the curr.record	++FS4
oRdd:ORDERKEYVAL	Returns the value of the current index key	++FS4
oRdd:ORDESCOPE()	Sets a boundary for scoping	++FS4
oRdd:ORDERSKIPUNIQUE()	Moves to next/previous key regardless Unique	++FS4
oRdd:ORDERTOPSCOPE	Controls the index visibility	++FS4
oRdd:PACK()	Removes all records marked for deletion	++FS4
oRdd:QUICKFIELDGET()	Similar to FIELDGET()	++FS4
oRdd:QUICKFIELDPUT()	Similar to FIELDPUT()	++FS4
oRdd:RDDINFO()	Returns the information about the RDD	++FS4
oRdd:RDDNAME	Returns the name of the RDD	++FS4
oRdd:READONLY	Determines the read only flag	++FS4
oRdd:RECALL()	Reinstates the current delete mark	++FS4
oRdd:RECALLALL()	Reinstates delete marks for all records	++FS4
oRdd:RECCOUNT	Determines the number of records in the curr. database	++FS4
oRdd:RECNO	Determines or moves to the current/specified record	++FS4
oRdd:RECORDINFO()	Returns information about the specified record	++FS4
oRdd:RECSIZE	Returns the record size in bytes	++FS4
oRdd:REFRESH()	Rereads the current record	++FS4
oRdd:REINDEX()	Rebuilds all open indices	++FS4
oRdd:RELATION()	Retrieves the relation string	++FS4
oRdd:RELATIONOBJECT()	Retrieves the object of the relation	++FS4
oRdd:REPLACE()	Replaces one or several fields with a new value	++FS4
oRdd:RLOCK()	Locks the specified record	++FS4
oRdd:RLOCKLIST	Retrieves a list of locked records	++FS4
oRdd:RLOCKVERIFY()	Determines if a Rlock is safe	++FS4
oRdd:SCOPE	Determines the general server scope	++FS4
oRdd:SEEK()	Seeks for the first occurrence in index key	++FS4
oRdd:SEEKEVAL()	Searches for a substring in index keys	++FS4
oRdd:SETFILTER()	Sets a filter scope	++FS4
oRdd:SETINDEX()	Opens a index file	++FS4
oRdd:SETORDER()	Select an index/order from the list	++FS4
oRdd:SETORDERCONDITION()	Sets condition for indexing	++FS4
oRdd:SETRELATION()	Sets a relation to child data server	++FS4
oRdd:SHARED	Returns the shared flag	++FS4
oRdd:SKIP()	Move the record pointer to next/prev record	++FS4
oRdd:SORT()	Sorts the natural record order	++FS4
oRdd:SUM()	Calculates the sum of numeric expressions	++FS4
oRdd:TOTAL()	Summarizes records into other database	++FS4

oRdd:UNLOCK()	Releases a specified lock or all locks	++FS4
oRdd:UPDATE()	Updates the database from another data server	++FS4
oRdd:USED	Determines if the data server is usable	++FS4
oRdd:WHILEBLOCK	Sets/retrieves the general While block	++FS4
oRdd:ZAP()	Removes all records from the database	++FS4

Index of the FlagShip Extend System functions

_parc()	Returns a pointer to the FlagShip character string	FS3
_parclen()	Returns the length of the FlagShip character string	FS3
_parcsiz()	Returns the allocated length of the FlagShip string	FS3
_pards()	Returns a pointer to the string repres. of date variable	FS3
_parinfa()	Returns the type of an array element	FS3
_parinfo()	Returns the number of arguments or the argument type	FS3
_parl()	Returns the value of a FlagShip logical variable	FS3
_parnd()	Returns the "double" value of the FlagShip numeric var	FS3
_parni()	Returns the "int" value of the FlagShip numeric variable	FS3
_parnl()	Returns the "long" value of the FlagShip numeric var	FS3
_parscw()	Returns a copy of the "screen" variable	++FS3
_ret()	Puts NIL into the FlagShip return value	+FS3
_retc()	Copies a string into the FlagShip return value "char"	FS3
_retclen()	Copies a part of a string into the return FS value "char"	FS3
_retds()	Copies a date string into the FlagShip return value "date"	FS3
_retl()	Copies an integer into the FlagShip return value "logical"	FS3
_retnd()	Copies a double into the FlagShip return value "number"	FS3
_retni()	Copies an integer into the FlagShip return value "number"	FS3
_retnl()	Copies a long into the FlagShip return value "number"	FS3
_storc()	Copies a string into the FlagShip string variable	FS4
_storclen()	Copies a part of a string into the FlagShip char variable	FS4
_stords()	Copies a date string into the FlagShip date variable	FS4
_storl()	Copies an integer into the FlagShip logical variable	FS4
_stornd()	Copies a double into the FlagShip numeric variable	FS4
_storni()	Copies an integer into the FlagShip numeric variable	FS4
_stornl()	Copies a long integer into the FlagShip numeric variable	FS4
_xalloc()	Allocates memory on the heap	FS4
_xfree()	Frees allocated memory from the heap	FS4
_xgrab()	Allocates (and checks) memory on the heap	FS4
_xunlock()	Clipper: frees VM segments, no action in FS	*FS4
malloc(), free()	Standard heap functions are usable	++FS3
FSudfname()	Generates a FlagShip compatible function header	++FS3
FSinit()	Initializes FlagShip parameters	++FS3
FSreturn	Exits the C function back to the FlagShip program	++FS3
PCOUNT	Returns the number of parameters received	FS3
ALENGTH()	Evaluates the length of a FlagShip array parameter	FS3
ISCHAR()	Checks if the argument is of type "character"	FS3
ISNUM()	Checks if the argument is of type "numeric"	FS3
ISLOG()	Checks if the argument is of type "logical"	FS3
ISDATE()	Checks if the argument is of type "date"	FS3
ISMEMO()	Checks if the argument is a memo field	FS3
ISARRAY()	Checks if the argument is an array	FS3
ISSCREEN()	Checks if the argument is of type "screen"	++FS3

Index of the FlagShip Open C API System

#Cinline	Begin of the C code within the .prg file	++FS4
#endCinline	End of the C code within the .prg file	++FS4
IS_VAR_ARR()	Checks if the variable is an array	++FS4
IS_VAR_BLK()	Checks if the variable is a code block	++FS4
IS_VAR_BYREF()	Checks if the variable is passed by reference	++FS4
IS_VAR_CHR()	Checks if the variable is a string	++FS4
IS_VAR_DATE()	Checks if the variable is a date	++FS4
IS_VAR_EE()	Checks if var1 == var2	++FS4
IS_VAR_EMPTY()	Checks if the variable is empty or NIL	++FS4
IS_VAR_EQ()	Checks if var1 = var2	++FS4
IS_VAR_FALSE()	Checks if the logical variable is false	++FS4
IS_VAR_FIELD()	Checks if the variable is a .dbf field	++FS4
IS_VAR_FP()	Checks if the variable is float numeric	++FS4
IS_VAR_GE()	Checks if var1 >= var2	++FS4
IS_VAR_GT()	Checks if var1 > var2	++FS4
IS_VAR_INT()	Checks if the variable is integer numeric	++FS4
IS_VAR_LE()	Checks if var1 <= var2	++FS4
IS_VAR_LOG()	Checks if the variable is a boolean	++FS4
IS_VAR_LT()	Checks if var1 < var2	++FS4
IS_VAR_NE()	Checks if var1 != var2	++FS4
IS_VAR_NIL()	Checks if the variable is undefined, NIL	++FS4
IS_VAR_NUM()	Checks if the variable is numeric	++FS4
IS_VAR_OBJ()	Checks if the variable is an object	++FS4
IS_VAR_SCR()	Checks if the variable is a screen type	++FS4
IS_VAR_SPECIAL()	Checks if the variable is a special C type	++FS4
IS_VAR_STD()	Checks if the variable is of standard type	++FS4
IS_VAR_TRUE()	Checks if the logical variable is true	++FS4
IS_VAR_TRUE_BLK()	Checks if the code block evaluates to true	++FS4
OBJ_ACCEXEC()	Executes an access method or access to an instance	++FS4
OBJ_ASSEXEC()	Executes an assign method or assign to an instance	++FS4
OBJ_DECL_METH()	Declares a function header of method	++FS4
OBJ_DECL_METHACCESS()	Declares a function header of access method	++FS4
OBJ_DECL_METHASSIGN()	Declares a function header of access method	++FS4
OBJ_METHEXEC()	Executes a method of a class	++FS4
SET_VAR_ADD()	Adds two FlagShip variables	++FS4
SET_VAR_BLOCK()	Assigns a function to a code block variable	++FS4
SET_VAR_CHR()	Copies a string into a FlagShip variable	++FS4
SET_VAR_CHRLEN()	Copies a string of spec.length into a FS variable	++FS4
SET_VAR_COPY()	Copies one FS variable into another one	++FS4
SET_VAR_DATE()	Stores date equivalence to a FS variable	++FS4
SET_VAR_INT()	Stores long integer number to a FS num variable	++FS4
SET_VAR_LOG()	Stores boolean to a FS logical variable	++FS4
SET_VAR_LOWER()	Converts FlagShip char variable to lower case	++FS4
SET_VAR_MACRO()	Evaluates a macro expression	++FS4
SET_VAR_NIL()	Resets a FlagShip variable to NIL	++FS4
SET_VAR_NUM()	Stores float number to a FS num variable	++FS4
SET_VAR_NUMDECI()	Stores float number to a FS variable, sets decimals	++FS4

SET_VAR_SCR()	Stores screen contents into a FS variable	++FS4
SET_VAR_SPECIAL()	Stores user defined pointer into a FS variable	++FS4
SET_VAR_UPPER()	Converts FlagShip char variable to upper case	++FS4
UDF_DECL()	Declares the header of user defined function	++FS4
UDF_EXEC()	Executes a standard or user defined function	++FS4
UDF_PROT()	Prototypes a user defined function	++FS4
VAR_ARRELEM()	Access to an FS array element	++FS4
VAR_BLOCK()	Returns the address of code block	++FS4
VAR_BLOCK_COMPILE()	Creates a code block variable from a string	++FS4
VAR_BLOCK_EVAL()	Evaluates a code block variable	++FS4
VAR_CHR()	Returns the stored string in FS variable	++FS4
VAR_DATE()	Returns the stored date in FS variable	++FS4
VAR_DEL_MARK()	Marks starting point for subsequent var deletion	++FS4
VAR_DEL_ONEVAR()	Deletes specified temp variable	++FS4
VAR_DEL_SINCE_MARK()	Deletes variable from starting point	++FS4
VAR_DELETE	Deletes all temp variables	++FS4
VAR_FPNUM()	Returns float number stored in FS int/float variable	++FS4
VAR_INT()	Returns the stored int number in FS IntVar variable	++FS4
VAR_INTNUM()	Returns long int stored in FS int/float variable	++FS4
VAR_ISDECI()	Returns the number of decimal digits displayed	++FS4
VAR_ISDIM()	Returns the size of array	++FS4
VAR_ISFLDPOS()	Determines the ordinal field position	++FS4
VAR_ISFLDWA()	Determines the working area of a field	++FS4
VAR_ISLEN()	Returns the size of string	++FS4
VAR_ISMODE()	Returns the additional status of a FS variable	++FS4
VAR_ISTYPE()	Returns the variable type	++FS4
VAR_LOG()	Returns the stored boolean in FS variable	++FS4
VAR_NAME_FIELD()	Determines the variable name from .prg part	++FS4
VAR_NAME_MEMVAR()	Determines the variable name from .prg part	++FS4
VAR_NAME_LOCAL()	Determines the variable name from .prg part	++FS4
VAR_NAME_LOCPAR()	Determines the variable name from .prg part	++FS4
VAR_NAME_STATIC()	Determines the variable name from .prg part	++FS4
VAR_NEW	Creates a new temp variable	++FS4
VAR_NEW_ARGS()	Creates an array of temp variables	++FS4
VAR_NEW_ARRAY()	Creates a new FlagShip array	++FS4
VAR_NEW_COPY()	Creates new temp var copying other one	++FS4
VAR_NEW_STATIC	Creates a new temp static variable	++FS4
VAR_NEW_STATIC_ARRAY()	Creates a new FlagShip static array	++FS4
VAR_NEW_STATIC_COPY()	Copies one static array copying to other	++FS4
VAR_NUM()	Returns the stored float number in FS variable	++FS4
VAR_OBJ()	Access to a FlagShip object element	++FS4
VAR_SCR()	Access to a screen contents stored in variable	++FS4
VAR_SPECIAL()	Access to the C pointer stored in FS variable	++FS4

FlagShip operators

%	Modulo of two numbers	(binary, mathem.)	FS3
*	Multiplication	(binary, mathem.)	FS3
** ^	Exponentiation	(binary, mathem.)	FS3
+	Addition, unary positive, concatenat.	(math, string)	FS3
•	Subtract., unary negative, concatenat.	(math, string)	FS3
/	Division	(binary, mathem.)	FS3
<> # !=	Not equal	(binary, relational)	FS3
$	Substring checking	(binary, relational)	FS3
.AND.	Logical AND	(binary, logical)	FS3
.NOT. !	Logical negation	(binary, logical)	FS3
.OR.	Logical OR	(binary, logical)	FS3
<	Lower then	(binary, relational)	FS3
<=	Lower or equal to	(binary, relational)	FS3
=	Equal to	(binary, relational)	FS3
==	Exactly equal to	(binary, relational)	FS3
>	Greater than	(binary, relational)	FS3
>=	Greater or equal to	(binary, relational)	FS3
=	Assignment	(binary, assignment)	FS3
:=	Inline assignment	(binary, assignment)	FS4
+=	Addition and assignment, concatenation	(binary, assignment)	FS4
-=	Subtraction and assignment	(binary, assignment)	FS4
*=	Multiplication and assignment	(binary, assignment)	FS4
/=	Division and assignment	(binary, assignment)	FS4
%=	Modulo and assignment	(binary, assignment)	FS4
**= ^=	Exponentiation and assignment	(binary, assignment)	FS4
++	Incrementation	(unary, mathem.)	FS4
--	Decrementation	(unary, mathem.)	FS4
&	Macro operation	(unary, special)	FS3
->	Alias operation	(unary, special)	FS3
@	Parameter passing by reference	(unary, special)	FS3
{ }	Array constants, code block definition	(special)	FS4
[ ]	Array element index	(special)	+FS3
( )	Functions or grouping indicator	(special)	FS3
:	Object variable or method	(special)	FS4

Precedence of operators

1.	Parentheses, special operators : `()` `[]` `{}` `&` `@`	+FS3
2.	++ pre-increment, `--` pre-decrement	FS4
3.	Mathematical and string operations (unary, exp, multip, addit)	+FS3
4.	Relational operations	FS3
5.	Logical operations (`.NOT.` `.AND.` `.OR.`)	FS3
6.	Assignments +FS3	
7.	Post-increment ++ , post-decrement `--`	FS4

Index of FlagShip Preprocessor Directives

#command	Specifies a command translation	FS4
#define	Defines a symbolic constant or parametriz. express.	FS4
#ifdef	Conditional compiling only if symbol defined	FS4
#ifndef	Conditional compiling only if symbol not defined	FS4
#endif	End of the #ifdef or #ifndef block	FS4
#include	Inserts the specified file	FS4
#translate	Specifies a translation	FS4
#undef	Clears a #define symbol	FS4
#xcommand	Specifies a literal command translation	FS4
#xtranslate	Specifies a literal translation	FS4
#Cinline	C inline statements in .prg code follows	++FS4
#endCinline	End of the C inline statements in .prg code	++FS4
#.....	any C preprocessor directive	++FS4
#define FlagShip	predefined to true by the FlagShip Compiler	++FS4
PUBLIC FlagShip	automatically sets .T. by the FlagShip Compiler	++FS3

These #define's are specified for compilation in the FS7config file depending on the FlagShip version and target system, or can be given as -D<value>. You may specify them in the source code to process different actions, e.g. in dependence on the current target platform. Note: the #define values are case sensitive, i.e. #ifdef FlagShip is not equivalent to #ifdef FLAGSHIP

```
#ifdef FlagShip
   ... FlagShip specific statements ...
#else
   ... Clipper specific statements ...
#endif

#ifdef FlagShip5
   ... Visual FlagShip (FlagShip 5 and 6) specific statements ...
#else
   ... FlagShip 4.48 specific statements ...
#endif

#ifdef FS_WIN32
   ... MS-Windows specific statements ...
#else
   ... Unix/Linux specific statements ...
#endif
```

Index of the FlagShip include files

(also the standard Clipper 5.x *.ch files are usable, except for the std.ch. The FlagShip's std.fh file may also be used with Clipper 5.x)

std.fh	Standard command translations	*FS4
stdfunct.fh	Prototypes of standard FlagShip functions	++FS4
stdclass.fh	Prototypes of all standard classes	++FS4
dataserv.fh	Prototypes of the DataServer class	++FS4
dbfidx.fh	Prototypes of the DbdIdx class	++FS4
getclass.fh	Prototypes of the GET class	++FS4
errclass.fh	Prototypes of the Error class	++FS4
tbrclass.fh	Prototypes of the TBrowse and TbColumn class	++FS4
achoice.fh	optional definitions for ACHOICE()	FS4
box.fh	optional definitions for BOX(), DISPBOX()	FS4
dbedit.fh	optional UDF definitions for DBEDIT()	FS4
dbstruct.fh	optional definitions for DBSTRUCT()	FS4
directry.fh	optional definitions for DIRECTORY()	FS4
error.fh	optional definitions for FSerror.prg	FS4
fileio.fh	optional definitions for low level file access	FS4
getexit.fh	definitions for getsys.prg	FS4
inkey.fh	optional definitions with INKEY() return codes	FS4
memoedit.fh	optional UDF definitions for MEMOEDIT()	FS4
set.fh	optional definitions for SET()	FS4
setcurs.fh	optional definitions for SETCURSOR()	FS4
simpleio.fh	optional definitions for OUTSTD()	*FS4
fspreset.fh	FS_SET() pre-definitions on program begin	++FS4
FlagShip.h	automatically used during the C compilation	++FS3
FSextend.h	needed for the FlagShip Extend System	++FS3
FSerrors.h	used for the FlagShip Error System	++FS3

FlagShip will try to infer the #include "file_name" if it is not able to find it as given, using the following search algorithm:

1. Look for the file name as given in #include "File.Ext"
   a. the current directory
   b. the path given by the -I switch (if any)
   c. the /usr/include directory
2. Repeat step a to c with "file.ext" (in lower cases)
3. Repeat step a to c with "FILE.EXT" (in capital letters)

To use the Clipper extensions in unmodified source code, link them in Linux e.g.

```
 ln /usr/local/Flag*/include/inkey.fh inkey.ch
```

but do not use Clipper's "std.ch" instead of "std.fh".

Summary of FlagShip Compiler Options

see details in section FSC

Syntax:

```
       FlagShip [options] parameter [parameter ...]
```

Help:

```
       FlagShip  (without parameters)
       FlagShip  -h  or: /h  or: --help
```

Parameters:

program.ext	List of the programs to be compiled/linked with FlagShip and/or C. Wildcards * and ? are allowed (the expansion is done by the Unix Shell or internally for MS-Windows). The 1st parameter is the name of main module (except with -M).
*.prg	FlagShip/Clipper/dBASE/Foxpro source program
*.frm	FlagShip/Clipper/dBASE/Foxpro format source program
*.c	by FS translated .prg into C or an external C program
*.o	object (Linux), by FlagShip and C translated programs
*.obj	object (Windows), by FlagShip and C translated programs
*.a	object libraries (Linux), created by "ar"
*.lib	object libraries (Windows), created by "lib" or "tlib"
*.so	dynamic libraries (Linux), created by "ld"
*.dll	dynamic libraries (Windows)

Options:

-32	Create 32-bit objects/executable. Default on 32bit systems
-64	Create 64-bit objects/executable. Default on 64bit systems
-a	Stop compilation after phase 1, produces .bp
-am	Use all PUBLICs, PRIVATEs and undeclared vars as MEMVAR
-b	Stop compilation after phase 2, produces .c
-c	Compile .prg and .c, produces .o, suppress linking
-C	Causes the applic to produce core dump on error
-d	Add debugger information, stop in debugger
-Dname[=value]	Define a symbol (and value) for the FS preprocessor and C
-delc	Delete intermediate .c file after creating object
-dyn	Link dynamically
-e name	Name the executable <name>, for Unix equivalent to -o switch
-et=value	Events process time in millisec (10..60000, default=100)
-f	Fast compilation, disable including default .fh files
-fox, -fxp	Handle also Fox array accesses with () for declared arrays

-g	Compile .c for debugger, set -nl, disable -s (strip)
-h, --help	Display this help only, don't compile
-i=name	Use #include <name>
-Ipath	Use <path> for #include's of *.h and *.fh
-io=a	Compile & link for auto i/o mode detected at startup (def)
-io=g|t|b	Compile & link for g=GUI or t=Terminal or b=Basic i/o mode
-iso	Translate source strings from ISO/ANSI to IBM-PC8/OEM
-lname	Use the library <name> (Unix/Linux only)
-Lpath	Use <path> for library search
-m	Compile modularly, don't look for external references
-Mname	Specify the <name> of the main module (procedure/function)
-mdi	Create MDI application instead of SDI (GUI mode only)
-na	Do not generate the automat. <prgname> procedure
-nc	Do not generate comments in the .c code
-nd	Do not generate info for the FlagShip debugger
-nD	Suppress the debugging information and trapping
-ne	Suppress automatic event trapping
-nI	Do not include "std.fh" automatically
-nI=file	Use the include "file" as standard instead of "std.fh"
-nl	Do not include .prg line statements in C code
-nL	Do not include C preproc. line statements in C code
-no	Do not optimize for size, but for speed
-nodelobj	Don't delete *.o or *.obj before compiling
-ns	Suppress visibility for procname(), procline() stack
-o name	Name the executable or object <name> instead of "a.out"
-outdel	Print deleted files with -v switch
-pm	Use all PUBLICs and PRIVATEs as MEMVAR
-q	Quiet mode, do not display line nums during compilation
-r=name	Specifies the name of the produced repository file
-rc	Add class prototypes into the repository file
-ro	The repository file overwrites the old one
-ru	Add UDF prototypes into the repository file
-stat	Link statically
-v	Verbose, display the ser#, compilation phases and options
-version	Display serial number & release of the FlagShip compiler
-w	Report undeclared variables as warnings, same as -w1
-w0	Disable warnings, report errors only
-w1	Report undeclared variables as warnings
-w2	Report untyped variables as warnings
-w3	Report unknown UDF return values and mismatched parameters
-w4	Report unresolvable CLASS references and late binding
-w5	Report prototyped but not declared methods and instances
-w6	Report automatic conversion of Fox array syntax
-Wc,-option	Pass <option> to the C compiler (cc)
-z	Do not use short .AND. evaluation

Examples:

```
   FlagShip myprog.prg

   FlagShip -m myprog.prg rest*.prg -omyprog
   FlagShip -na -nl -nd -c getsys.prg ; FlagShip -Mtest *.o
   FlagShip -w -I/usr/myincl -I/uu/fh -Mmymain [a-k]te*.prg *.o -otest
```

Configuration file FS8config

located in <FlagShip_dir>/etc/FS8config, where the <FlagShip_dir> depends on the installation; it default location is

for Unix/Linux: /usr/local/FlagShip8/etc/FS8config*
for MS-Windows: C:\FlagShip8\etc\FS8config*

In FlagShip8, there are two different configuration files: FS8config_32 for 32bit and FS8config_64 for creating 64bit executables. For 32bit cc (like Borland BCC compiler), there is only one configuration file. See further details in section FSC 1.4.2

FSDIR	Macro-holder definition for the <FlagShip_dir> path. Usually empty and determined from the compiler location
FSPATH	Directory including the FlagShip* executables. If not spe- cified, the path of the FlagShip command line invocation, and as last resort, the environment PATH is used.
CCPATH	Directory, including the CC executable or script. If not specified, the environment PATH is used.
CCNAME	Name of the C compiler, usually "cc".
CCDEBUG	Switch(es) to activate cc compilation in debug mode. If not specified, -g is used in Unix/Linux and "-Od -Zi" in Windows.
FSOPTIONS	Up to 32 FlagShip compiler options (see chapter FSC.1.3) separated by white space. Will precede the options given at the FlagShip command line.
MACRO1..MACRO9	are nine user-defined macros. If specified, the $(MACRO1), $(MACRO2) ... $(MACRO9) tag will be replaced by the content of this macro definition.
PREOPTIONS	Options passed to the <CCNAME> compiler, preceding the file name. Usually define's, includes, optimization and debugger info.
POSTOPTIONS	Options passed to cc following the file name. Usually linker options, libraries etc.
PREDYNAMIC	Similar to PREOPTIONS but used with the -dyn compiler switch
POSTDYNAMIC	Similar to POSTOPTIONS but used with the -dyn compiler switch
PRESTATIC	Similar to PREOPTIONS but used with the -stat compiler switch
POSTSTATIC	Similar to POSTOPTIONS but used with the -stat compiler switch
WINSYS_ATB	switch used for MS-Windows linker (in the POST* settings) when not compiled with -io=g switch. Default is -subsystem:CONSOLE
WINSYS_G	switch used for MS-Windows linker (in the POST* settings) when compiled with the -io=g switch. Default is -subsystem:WINDOWS

Invoking the executable

FlagShip for Unix/Linux:

a.out	invoke the Unix/Linux executable named a.out, search by PATH
./a.out	invoke the executable named a.out in the current directory
a.out par1 parN "my string"	Pass parameters to the executable, see also the CMD.PARAMETERS description
a.out -FSversion	Display the used FlagShip release
a.out -FSversion [param...]	Display the used FlagShip release
a.out -io=g [param...]	Invoke a.out in GUI mode, pass parameters
a.out -io=t [param...]	Invoke a.out in text mode, pass parameters
a.out -io=b [param...]	Invoke a.out in basic mode, pass parameters
newfswin a.out [par...]	Unix/Linux only: set environment for X/term, invoke executable named a.out (see RelNotes)
newfscons a.out [par...]	Unix/Linux only: set console TERM environment, invoke the executable named a.out and pass parameters <par...> into (see RelNotes)
newfsterm a.out [par...]	Unix/Linux only: set environment for remote terminal, invoke the executable named a.out (see sect REL for Release Notes)

FlagShip for MS-Windows:

name.exe [par1 ... parN]	Invoke MS-Windows executable in curr.dir, pass optional parameters separated by space
name.exe par1 "par2 with sp"	Invoke MS-Windows executable, pass two params
name -io=g [param...]	Invoke MS-Windows executable in GUI mode
name -io=t [param...]	Invoke MS-Windows executable in text mode
name -io=b [param...]	Invoke MS-Windows executable in basic mode
"c:\long path\name" par1 par2	Invoke MS-Windows executable, pass parameters

Environment variables

Environment variables for the FlagShip compiler:

FSCONFIG	Path of the configuration file "FSconfig" when not in the current or the /etc directory
PATH	Standard search path for executables including (drive and) directory where the FlagShip compiler modules resides (FlagShip, FlagShip_p and FlagShip_c), usually /usr/bin in Linux. On some BSD systems (like SUN) the PATH must include /usr/5bin as the first entry.
TMPDIR	Path used to create temporary files for the FlagShip and the C compiler, linker and librarian. If not set, the default /tmp directory is used. Make sure to have full access rights (rwx) to this directory and have enough free space available.

Environment variables during the execution of the application (export them and check using "env" or "printenv" in Linux or "set" in Windows):

FLAGSHIP_DIR	Path used by some tools like newfs*, distribute* etc.
PATH	Standard Unix path including the directory containing the actual executable and/or "::" for the actual directory. Also needed also for the correct execution of some functions like DIRECTORY() or ADIR(). On some BSD systems (like SUN) the PATH must include /usr/5bin as the first entry. Example for BSD: PATH=/usr/5bin:$PATH::$HOME ; export PATH
TERM	Definition of the currently used terminal (preferably the FSxxx one, see Predefined Terminals and system dependent notes). Example: TERM=FSansi ; export TERM
TERMINFO	Path with the terminfo definition (FStinfo.src), if not in- stalled in /usr/lib/terminfo.
FSTERMINFO	Optional, FlagShip specific. Equivalent to TERMINFO, but may coexist with it. If specified, only FSTERMINFO is used during the Curses initialization in the FlagShip application.
LANG	Usually not set, sometimes set to Unicode (UTF8). On difficulties with decimal point/comma conversion (see fscheck.prg), or with semi-graphic in terminal i/o mode on X11, set it to LANG=en_EN.ISO-8859-1 ; export LANG.
LINES, COLUMNS	Optional, overrides the terminfo specification of lines# and cols#
TZ	Time conversion, mostly set for specific time zone. Used for function TIME(), DATE() etc. See also "man environ" and "man tz".
FSOUTPUT	FlagShip specific. Path of the standard spool printer file, if the current directory should not to be used. Example: export FSOUTPUT=/usr/spool
x_FSDRIVE	FlagShip specific. Substitution of a DOS drive letters with an Unix directory. "x" represents A...Z drive letter, e.g.: export C_FSDRIVE=/usr/data
SCRMAP	FlagShip specific. Path for the language dependent sorting tables for INDEX and messages, if FS_SET ("loadlang") is used and the required file is not in the current directory. Example: SCRMAP=/usr/data ; export SCRMAP
TMPDIR	Path used to create temporary files for the FlagShip and the C compiler, linker and librarian. If not set, the default /tmp directory is used. Make sure

	to have full access rights (rwx) to this directory and have enough free space available.
FSPACKDIR	If specified, path used to create temporary files during the PACK execution.
LD_RUN_PATH	Path specifying where the executable should search for dynamic libraries (e.g. libFlagShip*.so) if not installed in the default /usr/lib directory.

Environment variables used in GUI debugger mode:

FSDEBUG_AUTO	Enables auto save/restore debugger status, see FSC.5.1
FSDEBUG_COMPILER	Path of the FlagShip executables (compiler) used by the GUI source-code debugger. The default setting is <FlagShip_dir>/bin (where <FlagShip_dir> = /usr/local/FlagShip8 for Linux and C:\FlagShip8 for MS-Windows)
FSDEBUG_INCLUDE	Path containing the std.fh file for GUI source-code debugger. The default setting is <FlagShip_dir>/include
FSDEBUG_SOURCE	Path containing the .prg source files used by the GUI source debugger. The default setting is the current directory. You may define several paths separated in Unix/Linux by colon(`:`) or by semicolon(`;`) in MS-Windows.
FSDEBUG_TMPREAD	Path and file name containing a pipe used by the GUI source-code debugger. The default setting is /tmp/fs<pid>.dbgin
FSDEBUG_TMPWRITE	Path and file name used by the GUI source debugger. The default setting is /tmp/fs<pid>.dbgout

File extensions and compatibility between DOS and Unix

.prg .frm .c .ch	The program/source files are fully compatible in both directions. For the transfer, both "text" and "binary" modes (with or without the CR/LF conversion) are usable. You may use also FlagShip or Unix tools (FSload, dos2unix, unix2dos, doscp etc.) for the transfer and conversion.
.obj .lib	The already translated programs into native (machine) code (object code and libraries) from 16bit Clipper/DOS are not usable by FlagShip in 32bit mode. The sources must be recompiled (using FlagShip or cc) on the target system again.
.dbf .dbt	Fully binary compatible. The transfer must always be done in binary mode, without the CR/LF conversion.
.dbv	FlagShip specific, binary, compatible to Windows/Linux
.fpt	FoxPro memo file, binary, compatible to FlagShip
.mem	Fully binary compatible, except for the screen variables. The transfer must always be done in binary mode, without the CR/LF conversion. You may suppress the storing of arrays and screen variables from FlagShip using FS_SET("memcompat", .T.).
.idx .ntx .ndx	FlagShip uses an optimized index structure for 32bit with the .idx extension. The Clipper's and dBase/FoxBase index files are not compatible. The same named .idx files from FoxPro are incompatible. FlagShip's .idx files are cross-compatible to different Unix and MS- Windows platforms. See also INDEXEXT() and FS_SET("translext").
.txt	FlagShip reads and converts DOS text files with the CR/LF line termination, and creates Unix format text files with the LF line terminator.
.exe	MS-Windows executable, named same as the main module or set by the -o or -e switch. You may rename it to any other *.exe name.
a.out	Default name of executable in Unix/Linux. You may set any name with and without extension by using the -o or -e switch, or simply rename the file a.out to anything else (file must have "rx" permission).

Notable differences of FlagShip to CA/Clipper and MS-DOS

- Unix file names are case sensitive, but FlagShip will optionally convert them automatically to the Unix convention (see example below).
- Clipper's .NTX indices cannot be used on Unix, FlagShip's .idx indices must be created on the target system using INDEX ON...TO.
- MS-DOS object files and libraries cannot be used for Unix, the .prg or .c sources must be recompiled on the target system.
- Note differences in system commands, if used in RUN.
- FlagShip produces spooled printer output by default. To use directly output to the device driver, use e.g. SET PRINTER TO /dev/lp0 in Linux or SET PRINTER TO PRN: in Windows. Printing to USB and network printers is supported in GUI mode by PrintGUI()
- Screen variables of type "S" are used in FlagShip for SAVESCREEN() variables, instead of the var type "C" in Clipper. Converting functions SCREEN2CHR() and CHR2SCREEN() are available to store them into dbf or dbt fields.
- Binary 0 (represented by CHR(0)) acts as the string terminator by convention in the C programming language. If necessary you can embed CHR(0) in a FlagShip character variable by means of invocation of the function FS_SET("zerobyte").
- File attributes (used e.g. in the ADIR() or DIRECTORY() function) of Unix (drwxrwxrwx) differs to the MS-DOS ones (a,d,s,h,v).
- For portable programs, omit the [Alt]+[key] combinations, since they are not available on most older Unix terminals.
- Clipper's .clp and .lnk files are not supported, since FlagShip's com- mand line entry is more powerful. Nevertheless, FlagShip also supports batch and scrips files. The usage of make tool is possible, the "fsmake" tool (in <FlagShip_dir>/tools) creates Makefile template semi-automatically.
- Additional settings using FS_SET(...) are available.

normally no other modification necessary:

1. **Include** the following statements into the **main .prg module**:

   ```
   *** main module, remains fully compatible to Clipper 5.x ***
   *
   #ifdef FlagShip
      #include "fspreset.fh"             // = all following definit
   #endif
   ```

   - or - insert your preferred definitions as needed:

   ```
   #ifdef FlagShip                         // automatically defined i
      FS_SET ("lower", .T.)                // convert files to lower
      FS_SET ("pathlower", .T.)            // paths and drives to low
      FS_SET ("translext", "ntx", "idx") // search for .idx instead
      IF GETENV ("C_FSDRIVE") == ""        // only if drive letter C:
         ? "set C_FSDRIVE env.variable first"
         QUIT
      ENDIF
   #endif
   ```

   That's all there is to it. The rest of the application (all .prg modules) remain unchanged, e.g.

   ```
   SET PATH TO C:\test\Data;..\Xyz\ABC
   SET DEFAULT TO ("D:\other\Data")
   IF .not. FILE("XYZ.NTX") ...
   ```

2. **Set the proper terminal** (preferably the FSxxx one) as listed in the "Predefined Terminals" or in the additional system info pages (REF):

   ```
   $ TERM=FSansi ; export TERM
   ```

3. Set Unix **environment variable**(s) prior to the execution of the a.out, if drive letters are used:

   ```
   $ D_FSDRIVE=/usr/data2 ; export D_FSDRIVE   (if D: or d: drive l
   $ C_FSDRIVE=/usr/data1 ; export C_FSDRIVE   (if C: or c: drive l
   $ D_FSDRIVE=/usr/data2 ; export D_FSDRIVE   (if D: or d: drive l
   ```

4. Check other environment variables and the additional system info pages.
5. Read also the First-steps-*.pdf booklet located in <FlagShip_dir> directory.

MULTISOFT

multisoft Datentechnik
Schönaustr. 7
D-84036 Landshut

http://www.fship.com
sales@multisoft.de
support@flagship.de