

Sheet1

NAME,C,20	TYPE,C,10	DEF,C,12
ADEL()	ARRAYS	Function
ADIR()	ARRAYS	Function
AFILL()	ARRAYS	Function
AINS()	ARRAYS	Function
ARRREST()	ARRAYS	UDF
ARRSAVE()	ARRAYS	UDF
ARRSORT()	ARRAYS	UDF
ASCAN()	ARRAYS	UDF
ATYPE()	ARRAYS	UDF
CREACHAR	ARRAYS	UDF
DECLARE	ARRAYS	Command
FREECHAR	ARRAYS	Procedure
PASSARR	ARRAYS	Procedure
PASSCHAR	ARRAYS	Procedure
READARRC	ARRAYS	UDF
READARRN	ARRAYS	Procedure
RETARR	ARRAYS	Procedure
RETCAR	ARRAYS	Procedure
SORTARRN	ARRAYS	Procedure
WRITARRC	ARRAYS	Procedure
WRITARRN	ARRAYS	Procedure
DEPDB()	BUSINESS	UDF
DEPSL()	BUSINESS	UDF
DEPVALDB()	BUSINESS	UDF
DEPVALSL()	BUSINESS	UDF
EFFYIELD()	BUSINESS	UDF
EOQ()	BUSINESS	UDF
INCTIME()	BUSINESS	UDF
TOMONEY()	BUSINESS	UDF
ALLTRIM()	CHARACTER	UDF
ASC()	CHARACTER	Function
AT()	CHARACTER	Function
ATLAST()	CHARACTER	UDF
ATNEXT()	CHARACTER	UDF
CAPFIRST()	CHARACTER	UDF
CENTER()	CHARACTER	UDF
CHR()	CHARACTER	Function
CHRCOUNT()	CHARACTER	UDF
CHRFOUND()	CHARACTER	UDF
CHRSWAP()	CHARACTER	UDF
CTOD()	DATE	Function
DECRYPT()	CHARACTER	UDF
DTOC()	DATE	Function
ENCRYPT()	CHARACTER	UDF
EXPAND	CHARACTER	UDF
ISALPHA()	CHARACTER	Function
ISLOWER()	CHARACTER	Function
ISUPPER()	CHARACTER	Function

LEADCHAR()	CHARACTER	UDF
LEFT()	CHARACTER	Function
LEN()	CHARACTER	Function
LJUST()	CHARACTER	UDF
LOWER()	CHARACTER	Function
LTRIM()	CHARACTER	Function
REPLICATE()	SCREEN	Function
RIGHT()	CHARACTER	Function
RJUST()	CHARACTER	UDF
RTRIM()	CHARACTER	Function
SPACE()	CHARACTER	Function
STUFF()	CHARACTER	Function
SUBSTR()	CHARACTER	Function
TRIM()	CHARACTER	Function
UPPER()	CHARACTER	Function
VAL()	CHARACTER	Function
WRAP()	CHARACTER	UDF
ALIAS()	DATABASE	Function
APPEND	DATABASE	Command
APPEND BLANK	DATABASE	Command
APPEND FROM	DATABASE	DATABASE
APPEND FROM	DATABASE	DATABASE
BOF()	DATABASE	Function
BROWSE	DATABASE	Interactive
CHANGE	DATABASE	Interactive
CLOSE	SYSTEM	Command
CONTINUE	MISC	Command
COPY FILE	DATABASE	Command
COUNT	DATABASE	Command
CREATE FROM	DATABASE	Command
CREATE	DATABASE	Command
CREATE QUERY	DATABASE	Command
CREATE VIEW	DATABASE	Command
DELETE	DATABASE	Command
DELETED()	DATABASE	Function
DISPLAY	DATABASE	Command
DISPLAY STRUCTURE	DATABASE	Interactive
EDIT	DATABASE	Interactive
EOF()	DATABASE	Function
ERASE	DISK	Command
EXPORT	DATABASE	Interactive
FCOUNT()	DATABASE	Function
FIELD()	DATABASE	Function
FIELDNAME()	DATABASE	DATABASE Fun
FIND	MISC	Command
FLDCOUNT()	DATABASE	UDF
FOUND()	MISC	Function
GETAREA()	DATABASE	UDF
GO	DATABASE	Command

HEADER()	DATABASE	UDF
IMPORT	DATABASE	Command
INDEX	DATABASE	INDEX Comman
INDEXKEY()	DATABASE	Function
JOIN	DATABASE	Command
LABEL FORM	MISC	Command
LASTREC()	DATABASE	Function
LIST	DATABASE	Command
LOCATE	DATABASE	Command
LUPDATE()	DATABASE	Function
MODIFY QUERY	DATABASE	Interactive
MODIFY STRUCTURE	DATABASE	Interactive
MODIFY VIEW	DATABASE	Interactive
NDX()	DATABASE	Function
NDXSIZE()	DATABASE	INDEX UDF
NEXTAREA()	DATABASE	UDF
NTXSIZE()	DATABASE	UDF
PACK	DATABASE	Command
RECALL	DATABASE	Command
RECNO()	DATABASE	Function
RECOUNT()	DATABASE	Function
RECSIZE()	DATABASE	Function
REINDEX	DATABASE	Command
RENAME	DISK	Command
REPLACE	DATABASE	DATABASE Com
SEEK	DATABASE	Command
SELECT	DATABASE	Command
SET CATALOG TO	DATABASE	Command
SET INDEX TO	DATABASE	Command
SET ORDER	DATABASE	Command
SET UNIQUE	DATABASE	Command
SKIP	DATABASE	CLIPPER Comm
SKIP	DATABASE	CLIPPER Comm
SORT	DATABASE	DATABASE
SORT	DATABASE	DATABASE
STREET()	CHARACTER	UDF
TOTAL TO	DATABASE	Command
UPDATE	DATABASE	Command
USE	DATABASE	Command
ZAP	DATABASE	Command
BOQTR()	DATE	UDF
BOW()	DATE	UDF
COW()	DATE	Function
CDTOS()	DATE	UDF
CMONTH()	DATE	Function
CMONTHN()	DATE	UDF
DAY()	DATE	Function
DOW()	DATE	Function
DTOS()	DATE	Function

EQQTR()	DATE	UDF
EOW()	DATE	UDF
INMONTHS()	DATE	UDF
INWEEKS()	DATE	UDF
INWKDAYS()	DATE	UDF
ISDSDATE()	DATE	UDF
ISLEAP()	DATE	UDF
ISWKEND()	DATE	UDF
LASTDAY()	DATE	UDF
MAKEDATE()	DATE	UDF
MONTH()	DATE	Function
NTOD()	DATE	UDF
QUARTER()	DATE	UDF
STOD()	DATE	UDF
WEEKDAYS()	DATE	UDF
YEAR()	DATE	Function
DISPLAY HISTORY	DEBUG	Interactive
INT3	DEBUG	Procedure
LIST HISTORY	DEBUG	Interactive
MEMDUMP	DEBUG	Procedure
RESUME	PROGRAM	Interactive
SET DOHISTORY	DEBUG	Interactive
SET HISTORY TO	DEBUG	Interactive
CURDIR()	DISK	UDF
CURDRIVE()	DISK	UDF
DIRMAKE()	DISK	UDF
DISKSPACE()	DISK	Function
FILE()	DISK	Function
ISDRIVE()	DISK	UDF
ISFIXED()	DISK	UDF
DISPLAY STATUS	DEBUG	Interactive
SELECT()	DATABASE	Function
SET ALTERNATE TO	DISK	Command
SET ALTERNATE	DATABASE	Command
SET BELL	SYSTEM	Command
SET CARRY	MISC	Command
SET CATALOG	DATABASE	Command
SET CENTURY	DATE	Command
SET COLOR	SCREEN	Command
SET COLOR TO	SCREEN	Command
SET CONFIRM	SYSTEM	Command
SET CONSOLE	SCREEN	Command
SET DATE	DATE	Command
SET DEBUG	DEBUG	Interactive
SET DECIMALS TO	NUMERIC	Command
SET DEFAULT TO	DISK	Command
SET DELETED	DATABASE	Command
SET DELIMITERS TO	SCREEN	Command
SET DELIMITERS	SCREEN	Command

SET DEVICE TO	SYSTEM	Command
SET ECHO	SCREEN	Interactive
SET ESCAPE	PROGRAM	Command
SET ESCAPE	PROGRAM	Command
SET EXACT	DATABASE	Command
SET FIELDS TO	DATABASE	Command
SET FIELDS	DATABASE	Command
SET FILTER TO	DATABASE	CLIPPER Comm
SET FIXED	NUMERIC	Command
SET FORMAT TO	SCREEN	Command
SET FORMAT TO	SCREEN	Command
SET FUNCTION	SYSTEM	Command
SET HEADING	SCREEN	Command
SET HELP	DEBUG	Command
SET HISTORY	DEBUG	Interactive
SET INTENSITY	SCREEN	Command
SET KEY	SYSTEM	Command
SET MARGIN TO	PRINTER	Command
SET MEMOWIDTH TO	MEMO	Command
SET MENUS	SCREEN	Command
SET MESSAGE	SCREEN	Command
SET ORDER TO	DATABASE	ENVIRON
SET PATH TO	DISK	Command
SET PRINT	PRINTER	Command
SET PRINTER	NETWORK	Command
SET PROCEDURE TO	PROGRAM	Command
SET RELATION TO	DATABASE	Command
SET SAFETY	DATABASE	Command
SET SCOREBOARD	SCREEN	Command
SET STEP	DEBUG	Interactive
SET TALK	DEBUG	Interactive
SET TITLE	DATABASE	Command
SET TYPEHEAD	SYSTEM	Command
SET VIEW TO	DATABASE	Command
LINES()	MEMO	UDF
TABSTRIP()	MEMO	UDF
WPSTRIP()	MEMO	UDF
ACCEPT	SCREEN	Command
AVERAGE	NUMERIC	Command
CLEAR ALL	SYSTEM	Command
CLEAR MEMORY	MISC	Command
DISPLAY MEMORY	DEBUG	Interactive
READ	DATABASE	Command
RELEASE	SYSTEM	Command
RESTORE FROM	DISK	Command
SAVE TO	DISK	MEMVAR Comma
STORE	PROGRAM	Command
WAIT	MISC	Command
ASSIST	MISC	Interactive

BLANK()	MISC	UDF
CREATE LABEL	MISC	Interactive
CREATE REPORT	MISC	Interactive
CREATE SCREEN	SCREEN	Interactive
FKLABEL()	MISC	Function
FKMAX()	MISC	Function
HARDCR()	MEMO	UDF
LABEL	MISC	Command
MODIFY COMMAND	PROGRAM	Interactive
MODIFY LABEL	DATABASE	Interactive
MODIFY REPORT	DATABASE	Interactive
MODIFY SCREEN	SCREEN	Interactive
PASSWORD()	CHARACTER	UDF
REPORT FORM	PRINTER	Command
STATUS()	SYSTEM	UDF
VERSION()	SYSTEM	MISC
ABS()	NUMERIC	Function
AVERAGE	NUMERIC	Function
BASE()	NUMERIC	UDF
BASE10()	NUMERIC	UDF
BIN()	NUMERIC	UDF
CEILING()	NUMERIC	UDF
DEC()	NUMERIC	UDF
EXP()	NUMERIC	Function
EXPONENT	NUMERIC	UDF
FACT	NUMERIC	UDF
FLR()	NUMERIC	UDF
HEX()	NUMERIC	UDF
INFINITY()	NUMERIC	UDF
INT()	NUMERIC	Function
LOG()	NUMERIC	Function
LOGNBASX()	NUMERIC	UDF
LOGTEN()	NUMERIC	UDF
MANTISSA()	NUMERIC	UDF
MAX()	NUMERIC	Function
MIN()	NUMERIC	Function
MOD()	NUMERIC	Function
POWER()	NUMERIC	UDF
ROOT()	NUMERIC	UDF
ROUND()	NUMERIC	Function
SQRT()	NUMERIC	Function
STR()	NUMERIC	Function
SUM	DATABASE	Command
EJECT	PRINTER	Command
ISPRINTER()	PRINTER	UDF
LPTSWAP	PRINTER	UDF
PRNSTATUS()	PRINTER	UDF
PRTSC	PRINTER	Procedure
SETPRC()	PRINTER	CLIPPER Func

TOF()	PRINTER	UDF
&	MISC	MACRO
?	SCREEN	Command
@...BOX	SCREEN	Command
@...PROMPT	SCREEN	Command
@...SAY	SCREEN	Command
@...TO	SCREEN	Command
@..SAY..GET	SCREEN	Command
CALL	MISC	Program
CANCEL	MISC	Command
DO	PROGRAM	Command
DO CASE	PROGRAM	Command
DO WHILE	PROGRAM	Command
EMPTY()	DATABASE	Function
ERROR()		PROGRAMMG
EXIT	MISC	Command
EXTERNAL	MISC	Command
FOR NEXT	MISC	Command
FUNCTION	MISC	Command
IF	PROGRAM	Command
IF()	PROGRAM	Function
IIF()	PROGRAM	Function
INKEY()	CHARACTER	Function
KEYBOARD	PROGRAM	Command
LASTKEY()	MISC	Function
LOOP	PROGRAM	Command
MEMOEDIT()	MEMO	UDF
MEMOREAD()	MEMO	UDF
MEMOTRAN()	MEMO	UDF
MEMOWRIT()	MEMO	UDF
MESSAGE()	MISC	Function
PARAMETERS	PROGRAM	Command
PCOL()	PRINTER	Function
PCOUNT()	PROGRAM	Function
PRIVATE	PROGRAM	Command
PROCEDURE	PROGRAM	Command
PROCLINE()	MISC	Function
PROCNAME()	MISC	Function
PROW()	PRINTER	Function
PUBLIC	PROGRAM	Command
QUIT	PROGRAM	Command
READKEY()	SYSTEM	Function
READVAR()	MISC	Function
RETRY	PROGRAM	PROGRAMMG
RETURN	PROGRAM	Command
SUSPEND	MISC	Command
TEXT	SCREEN	SCREEN Comma
TYPE()	DATABASE	Function
UPDATED()	DATABASE	Function

@..TO	SCREEN	Command
BIOSATTR()	SCREEN	UDF
CLEAR	SCREEN	Command
COL()	SCREEN	Function
CURSOR	SCREEN	SCREEN UDF
MENU	PROGRAM	Command
RESTORE SCREEN	SCREEN	Command
RESTSCR()	SCREEN	SCREEN UDF
ROW()	SCREEN	Function
SAVESCR()	SCREEN	UDF
SCRATTR()	SCREEN	UDF
SCROLL	SCREEN	UDF
SET COLOR TO	SCREEN	Command
TRANSFORM()	CHARACTER	Function
ALLOCATE()	SYSTEM	UDF
BUFFERS()	SYSTEM	UDF
CALL	MISC	SYSTEM
COUNTRY()	SYSTEM	UDF
DATE()	DATE	DATE/SYSTEM
DEALLOC()	SYSTEM	UDF
DIR	DISK	Command
DOSFUNC	SYSTEM	Process
DOSVERS()	SYSTEM	UDF
FILEATTR	SYSTEM	Process
FILEREAD()	SYSTEM	UDF
FILES()	SYSTEM	UDF
FILESIZE()	SYSTEM	UDF
FILEWRITE()	SYSTEM	UDF
FLOCK()	NETWORK	Command
GETENV()	SYSTEM	Function
ISCOLOR()	SYSTEM	Function
LASTDRIVE()	SYSTEM	UDF
LOAD	SYSTEM	Command
LOCK()	NETWORK	Function
MEMORY()	MISC	Function
NETERR()	NETWORK	Function
NETNAME()	NETWORK	Function
OFF()	SYSTEM	UDF
OFFSET	SYSTEM	Procedure
OS()	SYSTEM	Function
PEEK...()	SYSTEM	UDF
POKEBYTE()	SYSTEM	UDF
REG()	SYSTEM	UDF
RLOCK()	NETWORK	Function
ROMBIOS	SYSTEM	Procedure
SEG()	SYSTEM	UDF
SEGMENT	SYSTEM	Procedure
SET EXCLUSIVE	NETWORK	Command
TIME()	TIME	Function

Sheet1

UNLOCK	NETWORK	Command
USE...EXCLUSIVE	DATABASE	Command
DTOSS()	DATE	UDF
HTOSS()	TIME	UDF
ISTSTIME()	TIME	UDF
MAKETIME()	TIME	UDF
MTOSS()	TIME	UDF
SECONDS()	TIME	Function
SSTOD()	TIME	UDF
SSTOH()	TIME	UDF
SSTOM()	TIME	UDF
TIMESTR()	TIME	UDF
TSTOH()	TIME	UDF
TSTOM()	TIME	UDF
TSTOS()	TIME	UDF
COPY STRUCTURE	DATABASE	Command
DESCEND()	DATABASE	UDF
??	SCREEN	Command
==	OPERATOR	Command
!	OPERATOR	Command
&&	OPERATOR	Command
SOUNDEX()	MISC	UDF

SYNTAX,C,60
 ADEL(<array>,<pos>)
 ADIR(<skeleton>[,<array>])
 AFILL(<array>,<val>[,<start> [,<count>]])
 AINS(<array>,<pos>)
 ARRREST(<filespec>,<array name>)
 ARRSAVE(<filespec>,<array name>)
 ARRSORT(<source array>,<target array>)
 ASCAN(<array>, <val> [,<start> [,<count>]])
 ATYPE(<array name>[[<element number>]])
 CALL CREACHAR WITH <elements>,<size>,<expN3>
 DECLARE <array name> [<expN>][,<array list>]
 CALL FREECHAR
 CALL PASSARR WITH <element>,<position>
 CALL PASSCHAR WITH <element>
 CALL READARRC WITH <filespec>,<elements>,<size>,<error>
 CALL READARRN WITH <filespec>,<expN1>,<expN2>
 CALL RETARR WITH <variable>,<array position>
 CALL RETCHAR WITH <memvar>
 CALL SORTARRN WITH <array length>
 CALL WRITARRC WITH <filespec>,<elements>,<size>,<error>
 CALL WRITARRN WITH <filespec>,<elements>,<error>
 DEPD(<value>,<rate>,<periods>)
 DEPSL(<price>,<value>,<life>,<periods>)
 DEPVADB(<price>,<rate>,<periods>)
 DEPVALSL(<price>,<value>,<life>,<time>)
 EFFYIELD(<rate>, <compounded>)
 EOQ(<on hand>,<demand>,<cost>,<price>,<overhand>)
 INCTIME(<increase>,<rate>,<compounded>)
 TOMONEY(<expN>)
 ALLTRIM(<expC>)
 ASC(<expC>)
 AT(<expC>,<expC>)
 ATLAST(<character>,<string>)
 ATNEXT(<character>,<string>,<occurrence>)
 CAPFIRST(<expC>)
 CENTER(<string>, <width>)
 CHR(<expN>)
 CHRCOUNT(<character>,<string>)
 CHRFOUND(<string1>, <string2>)
 CHRSWAP(<string>,<char1> [,<char2>])
 CTOD(<expC>)
 DECRYPT(<expC>, <password>)
 DTOC(<expD>)
 ENCRYPT(<expC>, <password>)
 EXPAND(<string>[,<spaces>])
 ISALPHA()
 ISLOWER(<expC>)
 ISUPPER(<expC>)

LEADCHAR(<string>,<char>)
 LEFT(<expC>,<expN>)
 LEN()
 LJUST(<expC>)
 LOWER(<expC>)
 LTRIM(<expC>)
 REPLICATE(<expC>,<expN>)
 RIGHT()
 RJUST(<expC>)
 RTRIM(<expC>)
 SPACE(<expN>)
 STUFF(<expC1>,<start>,<chars>,<expC2>)
 SUBSTR(<expC>,<start position>[,<number of chars>]
 TRIM(<expC>)
 UPPER(<expC>)
 VAL(<expC>)
 WRAP(<expC>,<max len>)
 ALIAS([<expN])
 APPEND [scope][<fieldlist>]FROM<file>[FOR/WHILE]<condition>
 APPEND BLANK
 APPEND FROM
 APPEND FROM
 BOF()
 BROWSE
 CHANGE FIELDS <field list>
 CLOSE [ALTERNATE/DATABASES/FORMAT/INDEXES/PROCEDURI
 CONTINUE
 COPY FILE <filename> TO <filename>
 COUNT [<scope>] [FOR/WHILE <condition>] TO <memvar>
 CREATE <new file> FROM <structure extended file>
 CREATE <.dbf file name>
 CREATE QUERY <filename>
 CREATE VIEW <filename>
 DELETE [<scope>] [FOR/WHILE <condition>]
 DELETED()
 DISPLAY [OFF] <field list> [FOR/WHILE <condition>]
 DISPLAY STRUCTURE
 EDIT [<scope>][FIELDS <list>][FOR/WHILE <condition>]
 EOF()
 ERASE <filename>
 EXPORT
 FCOUNT()
 FIELD()
 FIELDNAME(<expN>)
 FIND <character string>
 FLDCOUNT()
 FOUND()
 GETAREA(<workarea>, <function>)
 GO/GOTO BOTTOM/TOP/<expN>

HEADER()
IMPORT
INDEX ON <exp> TO <.ntx file name>
INDEXKEY(<expN>)
JOIN WITH <alias> TO <new file> FOR/WHILE <condition>
LABEL FORM <.lbl file> [SAMPLE][<scope>][TO PRINT]
LASTREC()
LIST [OFF][scope]<field list> [FOR/WHILE <condition>]
LOCATE [<scope>] [FOR/WHILE <condition>]
LUPDATE()
MODIFY QUERY
MODIFY STRUCTURE
MODIFY VIEW
NDX()
NDXSIZE(<key size>,<number of records>)
NEXTAREA()
NTXSIZE(<key size>,<number of records>)
PACK
RECALL [<scope>] [FOR/WHILE <condition>]
RECNO()
RECOUNT()
RECSIZE()
REINDEX
RENAME <current file name> TO <new file name>
REPLACE [<scope>]<field> WITH <exp>[FOR/WHILE <condition>]
SEEK <expression>
SELECT <work area/alias>
SET CATALOG TO [<.cat filename>/?]
SET INDEX TO <.ntx file list>
SET ORDER TO <expN>
SET UNIQUE on/OFF
SKIP <expN>
SKIP <expN> [ALIAS <selection>]
SORT
SORT
STREET(<street address>)
TOTAL TO <filename> ON <key> [scope] [FIELDS<fieldlist>]
UPDATE
USE
ZAP
BOQTR(<expD>)
BOW(<expD>)
CDOW(<expD>)
CDTOS(<expC date>, <format>)
CMONTH(<expD>)
CMONTHN(<month>)
DAY(<expD>)
DOW(<expD>)
DTOS(<expD>)

EOQTR(expD>)
 EOW(<expD>)
 INMONTHS(<start>,<end>)
 INWEEKS(<start>,<end>)
 INWKDAYS(<start>,<end>)
 ISDSDATE(<date string>)
 ISLEAP(<expD>/<year>)
 ISWKEND(<expD>)
 LASTDAY(<month>,<year>)
 MAKEDATE(<date string>[,<months>]
 MONTH()
 NTOD(<year>,<month>,<day>)
 QUARTER(<expD>)
 STOD(<date string>)
 WEEKDAYS(<date>,<days>)
 YEAR(<expD>)
 DISPLAY HISTORY
 CALL INT3
 LIST HISTORY [LAST,<expN>][TO PRINT]
 CALL MEMDUMP WITH <exp>
 RESUME
 SET DOHISTORY on/OFF
 SET HISTORY TO <expN>
 CURDIR(<drive letter>)
 CURDRIVE()
 DIRMAKE(<directory>)
 DISKSPACE()
 FILE(<file name>)
 ISDRIVE(<drive>)
 ISFIXED(<drive>)
 DISPLAY STATUS
 SELECT()
 SET ALTERNATE TO <file name>
 SET ALTERNATE on/OFF
 SET BELL on/OFF
 SET CARRY on/OFF
 SET CATALOG ON/off
 SET CENTURY on/OFF
 SET COLOR ON/OFF
 SET COLOR TO [<standard>[,<enhanced>][,<border>][<background
 SET CONFIRM on/OFF
 SET CONSOLE ON/off
 SET DATE American/Ansi/British/Italian/French/German
 SET DEBUG on/OFF
 SET DECIMALS TO <expN>
 SET DEFAULT TO <drive>
 SET DELETED on/OFF
 SET DELIMITERS TO [<character string>][DEFAULT]
 SET DELIMITERS on/OFF

SET DEVICE TO SCREEN/print
 SET ECHO on/OFF
 SET ESCAPE ON/off
 SET ESCAPE ON/off
 SET EXACT on/OFF
 SET FIELDS TO [<field list>/ALL]
 SET FIELDS on/OFF
 SET FILTER TO [<condition>]
 SET FIXED on/OFF
 SET FORMAT TO <.fmt file name>
 SET FORMAT TO [<filename>/?]
 SET FUNCTION <expN> TO <character string>
 SET HEADING ON/off
 SET HELP ON/off
 SET HISTORY ON/off
 SET INTENSITY ON/off
 SET KEY <expN> TO [<proc>]
 SET MARGIN TO <expN>
 SET MEMOWIDTH TO
 SET MENUS ON/off
 SET MESSAGE TO <expN>
 SET ORDER TO <expN>
 SET PATH TO [<path list>]
 SET PRINT on/OFF
 SET PRINTER TO <destination>
 SET PROCEDURE TO <procedure file name>
 SET RELATION TO <key exp> INTO <alias>
 SET SAFETY ON/off
 SET SCOREBOARD ON/off
 SET STEP on/OFF
 SET TALK ON/off
 SET TITLE ON/off
 SET TYPEHEAD TO <expN>
 SET VIEW TO <.vue filename>
 LINES(<expC>, <line length>)
 TABSTRIP(<expC> [,<spaces per tab>])
 WPSTRIP(<expC> [,<spaces per tab>])
 ACCEPT [<prompt>] TO <memvar>
 AVERAGE <fieldlist> TO <memvarlist> [FOR/WHILE <condition>]
 CLEAR ALL
 CLEAR MEMORY
 DISPLAY MEMORY
 READ
 RELEASE [<memvar>] [ALL [LIKE/EXCEPT <skeleton>]]
 RESTORE FROM <.mem file name> [ADDITIVE]
 SAVE TO <.mem filename> [ALL LIKE/EXCEPT <skeleton>]
 STORE
 WAIT
 ASSIST

BLANK(<exp>)
 CREATE LABEL <filename>
 CREATE REPORT <filename>
 CREATE SCREEN <filename>
 FKLABEL(<expN>)
 FKMAX()
 HARDCCR(<expC>)
 LABEL
 MODIFY COMMAND
 MODIFY LABEL
 MODIFY REPORT
 MODIFY SCREEN
 PASSWORD(<expC>)
 REPORT FORM <.frm filename> [<scope>][FOR/WHILE <condition>]
 STATUS(<SET parameter>)
 VERSION()
 ABS(<expN>)
 AVERAGE
 BASE(<base>,<decimal number>)
 BASE10(<base>,<base number>)
 BIN(<expN>)
 CEILING(<expN>)
 DEC(<hex string>)
 EXP()
 EXPONENT(<expN>)
 FACT(<expN>)
 FLR(<expN>)
 HEX(<expN>)
 INFINITY()
 INT(<expN>)
 LOG(<expN>)
 LOGNBASX(<number>,<base>)
 LOGTEN(<expN>)
 MANTISSA(<expN>)
 MAX(<expN1>,<expN2>)
 MIN(<expN>/<expD>,<expN>/<expD>)
 MOD(<expN1>,<expN2>)
 POWER(<number>,<power>)
 ROOT(<number>,<root>)
 ROUND(<expN>,<decimals>)
 SQRT(<expN>)
 STR(<expN>[,<length>[,<decimals>]])
 SUM <fieldlist> TO <memvar> [FOR/WHILE <condition>]
 EJECT
 ISPRINTER()
 CALL LPTSWAP
 PRNSTATUS()
 CALL PRTSC
 SETPRC(<expN>,<expN>)

TOF()
 &<character variable>
 ? <exp list>
 @ <top,left,bottom,right> BOX <string>
 @ <row,col> PROMPT <expC> [MESSAGE <expC>]
 @ <row,col> [SAY <exp> [PICTURE <clause>]] VALID <exp>
 @...TO
 @ <rol,col> [SAY <expC> [PICTURE <clause>]] [GET <exp>]
 CALL <process> WITH <exp> [WITH WORD(<expN>)]
 CANCEL
 DO
 DO CASE...CASE...[OTHERWISE]...ENDCASE
 DO WHILE...ENDDO
 EMPTY(<exp>)
 ERROR()
 EXIT
 EXTERNAL
 FOR <memvar> = <expN> TO <expN> [STEP <expN>]...NEXT
 FUNCTION <name>...RETURN <value>
 IF...[ELSE]...ENDIF
 IF(<exp1>,<exp2>,<exp3>)
 IIF(<expL>,<exp1>,<exp2>)
 INKEY(<expN>)
 KEYBOARD <expC>
 LASTKEY()
 LOOP
 MEMOEDIT(<expC>,[<expN>,<expN>,<expN>,<expN>],[<expL>])
 MEMOREAD(<filename>)
 MEMOTRAN(<expC>[,<hard-fix>][<soft-fix>])
 MEMOWRIT(<filename>,<expC>)
 MESSAGE()
 PARAMETERS <memvar list>
 PCOL()
 PCOUNT()
 PRIVATE <memory variable list>
 PROCEDURE <procedure name>
 PROCLINE()
 PROCNAME()
 PROW()
 PUBLIC <memory variable list>/Clipper
 QUIT
 READKEY()
 READVAR()
 RETRY
 RETURN
 SUSPEND
 TEXT...ENDTEXT
 TYPE()
 UPDATED()

@ <row,col> [CLEAR] TO <row2,col2> [DOUBLE]
BIOSATTR()
CLEAR
COL()
CALL CURSOR WITH <"on"/"off">
MENU TO <memvar>
RESTORE SCREEN [FROM <memvar>]
RESTSCR(<filespec> [,<memvar name>])
ROW()
SAVESCR(<filespec> [,<memvar name>])
SCRATTR()
CALL SCROLL WITH <TLrow>,<TLcol>,<BRrow>,<BRcol>,<lines>,U/L
SET COLOR TO [<standard>[,<enhanced>[,<border>]]]
TRANSFORM(<exp>,<picture>)
ALLOCATE(<bytes>)
BUFFERS()
CALL
COUNTRY()
DATE()
DEALLOC(<address>)
DIR [<drive>:] [<path>] [<skeleton>]
CALL DOSFUNC WITH <registers>, <flags>
DOSVERS()
CALL FILEATTR WITH <filespec>,<attributes>
FILEREAD(<filespec>)
FILES()
FILESIZE(<filespec>)
FILEWRITE(<filespec>,<text>)
FLOCK()
GETENV()
ISCOLOR()
LASTDRIVE()
LOAD <binary filename>[.<extension>]
LOCK()
MEMORY()
NETERR()
NETNAME()
OFF(<memvar name>)
CALL OFFSET WITH <memvar>,<return memvar>
OS()
PEEK...()
POKEBYTE()
REG(<register name>)
RLOCK()
CALL ROMBIOS WITH <interrupt>,<registers>,<flags>
SEG(<memvar name>)
CALL SEGMENT WITH <memvar>,<return memvar>
SET EXCLUSIVE ON/off
TIME()

Sheet1

UNLOCK [ALL]
USE <filename> [INDEX <index file> EXCLUSIVE[ALIAS <name>]]
DROSS(<days>)
HROSS(<hours>)
ISTSTIME(<time string>)
MAKETIME(<time string>)
MROSS(<minutes>)
SECONDS()
SSTOD(<seconds>)
SSTOH(<seconds>)
SSTOM(<seconds>)
TIMESTR(<seconds>)
TSTOH(<time string>)
TSTOM(<time string>)
TSTOS(<time string>)
COPY STRUCTURE TO <file> [FIELDS<field list>]
DESCEND(<expD>)
?? <exp list>
<exp> == <exp>
DO WHILE ! .T.
&& <text>
SOUNDEX(<expC>)

RETURNS1,C,60

<expL> .T. if restored properly, otherwise .F.

<expL> .T. if written correctly to disk, otherwise .F.

<expL> true if successful, false if not

<expC> uppercase letter indicating data type (same as the

<expN3> -1 if memory allocation error occurs

<expN3>-1 if read error occurs

<expN2> -1 if read error occurs

<expN3> -1 if write error occurs

<expN2> -1 if write error occurs

<expN> total depreciation over <expN3> periods

<expN> total depreciation over <expN4> periods

<expN> value of asset after specified time

<expN> value of an asset after specified time

Effective yield (effective rate of interest)

<expN> economic order quantity

<expN> number of years required for specified appreciation

<expC> of <expN> in words in the format

<expC> with leading and trailing blanks removed

Converts the first character of a string into ASCII code

Returns a number that shows the starting position of a

<expN> position of last occurrence of <character> in

<expN> position of <occurrence> of <characters> in <string>

<expC> with the first alpha character of each new word

<expC> of <string> with enough leading blanks added to

Return the ASCII representation of a number

<expN> number of times <character> occurs in <string>

<expL> true if each character in <string1> also occurs in

<expC> string with all occurrences of <char1> replaced

Converts a date that has been stored/entered as a

<expC> of <string> decoded according to <password>

Converts a date variable into a character variable

<expC> of <string> coded according to <password>

<expC> string with <spaces> between each character

Returns a logical .T. if specified character expression

Returns a logical .T. if the leftmost character of an

Returns a logical .T. if the leftmost character of string

<expC> with each leading blank space in <string> replaced
Returns the specified number of characters from the left
Returns the number of characters in the alphanumeric
<expC> with leading blanks moved to the end

Permits repeating a character expression a specified

<expC> with trailing blanks moved to the beginning
Removes trailing blanks from a character expression just
Creates a character string consisting of a specified

Extracts a specified part of a character string
TRIM removes trailing blanks from a character string
Converts lower case characters into upper case characters
Converts a string character into a numeric expression
<expN> position of last blank space before <max len>
Returns the alias of a SELECT area

Returns a logical .T. if the record pointer is at the

E]

Returns a value of true or false whether a record has been

Returns the number of fields in the current database

The name of any field within the currently active

<expN> integer number of fields in the currently selected
Returns .T. if a previous SEEK, FIND, LOCATE, or
Result of <function> in <workarea>

Return the KEY expression of an index where <expN> is the

Returns the number of records in the currently active

The date of the last update of the currently selected

<expN> maximum number of bytes in target index file that
Lowest available work area as <expN> integer
<expN> number of bytes in INDEX ON <key> expression

Permits access to the current record number of the selected

<expC> street name from <street address>

<expD> first day of the calendar quarter in which the
<expD> Monday of the week in which the parameter date
Returns the name of the day of the week from a date
<date string> of <expC date> in the format "YYYYMMDD"
Returns the name of the month from a date variable
<expC> name of month, lowercase with initial capital
Returns the numeric value of the day of the month from a
Returns a number that represents the day of the week from
Returns a string in YYYYMMDD format. This can be used to

Sheet1

<expD> last day of the calendar quarter in which the
<expD> Sunday of the week in which the parameter date falls
<expN> number of months between two dates
<expN> number of weeks between two dates
<expN> number of non_weekend days between two dates
<expL> true if <date string> is valid in format and contents
<expL> true if <date> or <year> are a leap year, otherwise
<expL> true if date is a Saturday or Sunday, otherwise
<expN> last day of the month
Valid <expD> of <date string>, even if <date string> is

<expD> date type of <year>, <month> and <day>
<expN> quarter of the year corresponding to <expD>
<expD> date type of <date string>
<expD> of <date> plus or minue <days> NOT counting week
Returns the complete numeric value of the year from a

<expC> optional letter of drive to examine
<expC> current drive as uppercase letter
<expL> true if directory exists, false if the directory
Returns an integer representing the number of bytes
FILE verifies the existence of a specified file
<expL> true if drive exists and is working, otherwise false
<expC> drive letter, uppercase or lowercase

<expN> number of lines required to display <expC> within
<expC> with tab characters replaced by spaces
<expC> with tab, line-feed, and carriage-return characters

Empty value of <exp>

<expC> contains the name of the function key whose number
<expN> represents the number of programmable function key
Returns the char expression with all soft carriage returns

<expN> in the range from eight to ten characters

<expC> or <expN> depending of SET parameter passed

Returns the absolute value of an numeric expression

<expC> value of <decimal number> in math format of <base>

<expN> decimal number (base 10) of <base number>

<expC> binary value of <expN>

Integer just above <expN>

<expN> decimal value of <expC> hex value, or zero if there

The exponential value of the numeric argument

Power of 2 used to represent <expN>

Product of all numbers between <expN> and 1

Integer just below <expN>

<expC> hexadecimal value of <expN>

The bit pattern used by the 8087/80287 to represent

Converts any numeric expression into an integer

LOG returns the natural logarithm of a given number

the logarithm of the <expN> to the base <expN2>

Logarithm of <expN> to the base 10

Number between 1 and 1.9999999... which is multiplied by

The higher value of two numerical expressions

Returns the lesser value of two numeric or date expressions

a number representing the remainder of <expN1> divided

<expN1> number being raised to <power>

<expN> which is the <expN2> root of <expN1>

Returns the rounded number to a specified number of

Returns the square root of a given positive number

Converts a numeric expression into a character string

a form feed (ASCII 12) to the printer

<expL> true if parallel printer is on-line and ready,

<expN> parallel printer status code

SETS the internal PROW() and PCOL() values to specified

<expL> true if printer is at top-of-form, otherwise false

Supports- [RANGE <exp,exp>]

Function evaluates TRUE if char is a null,num is 0, logical

Provides a means of creating a conditional expression with
Returns the ASCII code for a key pressed, INKEY(0) halts

Returns the ASCII code for the last key pressed, including

Edit memos, <expC>=memo,<expN>=4 coordinates of window

Returns the contents of a text file read from disk

Returns <expC> with carriage return/line feed replaced

Returns a logical True .T. if successful

The error message character string

Returns the number of parameters that have been passed

Returns the source code line number of the current

Returns the name of the current program or procedure

Returns the current row on the printer

Returns the name of the current GET/MENU variable or a

Returns the data type of a memory variable or database

Will return .T. if last READ changed any of the data

Returns the current column position of the cursor on the

<expC> false if file not found, otherwise true

Returns the current row location of the cursor

<expL> true if file successfully written, otherwise false

<expC> uppercase letter value of current SET COLOR

)

Returns a character string with the specified picture

<expC> 4-byte hex address of 1st segment. Null <expC> if

<expN> number of buffers on the BUFFERS= line in

<expN> code on COUNTRY= line in Config.sys

Returns the system date in the form mm/dd/yy

<expL> true if successful, otherwise false

<expC> is altered to contain contents of registers after

<expC> DOS version number

<expC> contents of disk text file

<expN> number of files on FILES= line in Config.sys

<expN> size of file in bytes

<expC> "DONE" is written successfully

Returns a logical .T. if the lock was successful

Returns a logical .T. if a color display is installed and

<expC> letter of the drive specified on LASTDRIVE= line

Returns a logical .T. if the lock is successful

Returns the available memory in K bytes. MEMORY(0) retur

Will return .F. if a USE,USE..EXCLUSIVE or APPEND BLAN

Returns the text of the computer name when Networking

<expC> variable's offset address as hexadecimal string

<expC> offset address of <memvar name> as a hexadecimal

Returns the name of the version of DOS under which the

<exp> value of data type at requested location

<expC> contents of specified register as hex string

Returns logical True (.T.) when the lock attempt was

<expC> variable's segment address as hexadecimal string

<expC> segment address of <memvar name> as a hexadeci

TIME returns the system time in the format hh:mm:ss

<expN> number of seconds that equals <days>
<expN> number of seconds that equal <hours>
<expC> time string in the format "HH:MM:SS", the same as
<expC> time string valid in format and contents
<expN> number of seconds that equal <minutes>
Returns system time as <seconds>.<hundredths>
<expN> number of days that equal <seconds>
<expN> number of hours that equal <seconds>
<expN> number of minutes that equal <seconds>
<expC> time string in the format "HH:MM:SS", the same as
<expN> number of hours that equal <time string>
<expN> number of minutes that equal <time string>
<expN> number of seconds that equal <time string>

Allows indexes of any field type to be created in Descending

<expC> soundex code as a four-character string in the

RETURNS2,C,60

TYPE() function but adds "A" for array type

"... DOLLAR(S) and <n> CENT(S)"

character string within another character string
<string>

uppercase and all other alpha characters lowercase
center in <width>

<string2> in any order, otherwise false

character string into a date string

begins with a alpha character
expression is a lowercase letter
is an uppercase, Otherwise, returns a logical .F.

by <char>
of the alphanumeric argument
argument

number of times

like the TRIM() function
number of blank

beginning of the file

deleted

database

database file
CONTINUE was successful

placement in the index list

database file

database file

could result from INDEX on <key> TO <TARGET>

file

parameter date falls
falls
variable

Null if <month> is less than 1 or greater than 12
date variable
a date variable
index on a date + character expression

parameter date falls

;

otherwise false

false

false

invalid date, plus or minus <months>

ends

date variable

could not be created

available on the default disk drive

<line length> columns

replaced by spaces

corresponds to the numeric expression
on the terminal
CHR(141), replaced with carriage returns CHR(13)

,

is an invalid character in <expC>

infinity (7F F0 00 00 00 00 00 00)

the power of 2 necessary to product <expN>

by <expN2>

decimal places

otherwise false

values

Supports- [VALID <exp>] [CLEAR]

is .F., and if a date is blank

out using the IF...ENDIF command structure
til a key is pressed, INKEY(n) waits n sec or til keypress

control keys

<expL>=an update flag

CHR(13)+CHR(10)=[hard-fix], CHR(141)+CHR(10)=[soft-fix]

from the command line or another procedure

program or procedure
that is being executed

null string if none if pending

field
in the associated GETs

screen

memory cannot be allocated
Config.sys

Zero (default) if COUNTRY not specified, -1 if no Config.sys

INT 21H function call

a logical .F. for a monochrome display
in Config.sys

the free pool space for data manipulation
fails in a network environment
If name never set, a null string is returned.

string
program is running

successful.

string

the TIME() function

returned by the TIME() function

order

format "A999"

PURPOSE,C,60

Shift array elements down one position from pos position
Fill an array with filenames from disk directory
Fill many array elements with one value
Shift array elements up one position below pos
Restore a Clipper array from a disk file
Save a Clipper array to a disk file
Sort a numeric,date or logical array in ascending order
Search for specific value within an array
Distinguish array names from memvar names, and determin
Allocate memory for a C character array
Creates a one-dimentional array, elements can be mixed.
Release memory allocated by CREACHAR for C character a
Pass a Clipper numeric,date or logical array to C for faster
Pass a Clipper character array to C for faster processing
Read a character array from a disk file into a C array
Read a numeric,date, or logical array from a disk file into
Return an array from C to Clipper
Return an array from C to Clipper
Sort a Clipper date/logical/numeric array in C
Write a C character array to a disk file
Write a C numeric, date, or logical array to a disk file
Computes depreciation of an item as a percentage of its
Compute depreciation of an item by straight-line method
Compute the depreciated value of an asset using declining-
Compute depreciated value of an asset using straight-line
Computes the effective rate of interest of a loan
Compute economic order quantity of an item
Determine number of years for an investment to increase by
Convert a numeric amount to words of currency
Remove blanks from both ends of a string

Determine rightmost position of character in string
Determines position of <n>th occurrence of char in string
Capitalize the first letter of each word, lowercase the
Center output without calculating its starting position

Determine how many times a specified character occurs in
Determine whether <string2> contains ALL the characters th
Replace one character with another throughout a string, or

Unprotect data that was protected with ENCRYPT()

Protect data from easily being read
Expand string with spaces for headers and titles
Evalute 1st character of an expression for alpha
Evaluate for lower case
Evaluate for upper case

Display a string with leading characters other than blanks
Substring selection from the left side
Determines the length of a character string
Left justify the contents of the string within that string
Converts upper case characters to lower case
Removes the leading blanks that result from the STR

SUBSTRING SELECTION FROM RIGHT SIDE
Right justify the contents of a string with a string

Replaces any part of a character string, without the need to

Break a long string for output within a specified line

Adds records from other files to database files
Adds blank records to the end of the database file
ADDS RECORDS FROM OTHER FILES
SUPPORTS FIELD LIST AS OPTIONAL ARGUMENT

Allows Edit, Delete, Add to a database
Edits specified data fields sequentially, with or
Closes the specified type of file
Positions pointer to next record meeting LOCATE conditions
Duplicates any kind of file
Tallies number of records that meet a specific condition
Create forms a new database file from CREATE command fi
Creates an empty structure extended file with 4 field
Creates a new query file (.QRY)
Creates a new view file (.VUE)
Flags the designated records for deletion

Displays records from the active database
DISPLAYS STRUCTURE OF A DATABASE FILE IN USE
Alters data fields in a database
END-OF-FILE
Delete the specified file from the directory
CREATES A PFS FILE FROM A DATABASE

NAMES OF FIELDS IN DATABASE FILES

Positions pointer to 1st record whose key matches string
Determine the number of fields in a database file

Obtain the result of a function from other than the
Positions pointer directly to specified record

Sheet1

Determine size of a database file header
CREATES DATABASE AND FORMAT FILES FROM PFS FILE
Causes database to appear to be sorted with specified key

Combines specified records and fields from two databases
Prints labels using the specified label form file

Lists the contents of a database file
Positions to a record that meets a specified condition
Show when a database file was last updated
MODIFIES A QUERY FILE
CREATES A BACKUP OF A DATABASE FILE OR MEMO FILE
MODIFIES A VIEW FILE
NAMES OF OPEN INDEX FILES
Determine potential max size of a dBASE 3 PLUS index file
Obtain the next available work area
Determine potential maximum size of a Clipper index file
Removes database records marked for deletion
Reinstates records marked for deletion

NO. OF RECORDS IN DATABASE
SIZE OF RECORD
Rebuilds existing active index files
Provides a new name to an existing file
Changes contents of data fields to specified values
Positions pointer to 1st record that matches expression
Select 0 selects first unused area
CREATES A NEW CATALOG
Opens the specified index file
Sets up any open index file as primary, without the need
Causes first/ALL record(s) with identical keys to be indexed
Positions pointer forward or backward
SKIP 0 Writes dbf buffer, ALIAS moves pointer unselected file
CREATES A SORTED VERSION OF THE ACTIVE DATABASE
CREATES A COPY OF A DATABASE FILE TO BE USED
Order a database by street name
Creates a summary database containing numeric fields
ALLOWS BATCH UPDATES OF A DATABASE
Specifies database to be used until another USE
Removes all records from the active database file
Determine the beginning of the current calendar quarter
Compute the beginning of the current week

Convert a character type date to a date string where it can

Get the character name of the month from a numeric month

Determine the end of the current calendar quarter
Compute the end of the current week
Determine the number of months between two dates
Determine the number of weeks between two dates
Determine the number of non-weekend days between two dates
Determine if a date string is valid
Determine if a date is in a leap year
Determine if a date is on a weekend
Determine date of the last day of the month
Make an invalid date string valid and (+) or (-) months
MONTH OF YEAR
Convert numeric date parameters to date type
QUARTER OF YEAR OF SPECIFIED DATE
Convert date string to date type
Add or subtract non-weekend days to or from dates

Displays the commands stored in HISTORY
Inserts a breakpoint into a program for machine-language
Lists commands that are stored in the HISTORY mode. It
Views memory byte-by-byte in hex and ASCII form
Causes a SUSPENDED program to resume execution
Determines whether commands from command files are received
Specifies the number of executed commands stored in HISTORY
Get name of current disk directory
Determine currently logged in drive
Create a directory if it does not already exist. Useful

Determine if a specified disk drive exists and is ready
Determine if a specified disk drive is a hard disk, useful
DISPLAYS CURRENT INFORMATION ABOUT THE FILES IN THE CURRENT DIRECTORY
Returns the numeric value of the currently selected area
Creates a file for saving output
Sends/DOES NOT SEND output to a file
Bell rings/DOES NOT RING during data entry operation
Writes/DOES NOT WRITE last record into an APPEND
ADDS/does not add files to the open catalog
Shows/DOES NOT SHOW the century in date displays
Switches between color and monochrome monitors on system
Sets screen displays attributes
Requires/DOES NOT REQUIRE the carriage return
SENDS/suspends all output to the screen
Determines the format for date expressions
Sends/DOES NOT SEND output of ECHO to the printer
Determines the minimum number of decimals displayed
Specifies the default drive for file operations
Does not display/DISPLAYS records marked for deletion
Specifies delimiters for fields and variable displays
Displays full-screen fields in normal video/REVERSE VIDEO

Sends @..SAY command output to the SCREEN/printer
Displays command lines from program on the screen and/or
The default is ON, where:
With SET ESCAPE ON, the ESC key interrupts the program
Requires/DOES NOT REQUIRE exact matches in char comp
Defines a list of fields that may be accessed in one or
If SET FIELDS is OFF all fields in database are available
Filters out database records not satisfying the condition
Fixes/DOES NOT FIX the number of decimals to be displaye
Opens a format file for data entry
Opens a format file for data entry
Used to re-program function 2 through 40
Field names DISPLAY/do not in LIST or DISPLAY
PROMPTS/does not prompt user for help when an error occu
Turns command history feature on and off
SETS ON/sets off the reverse video in full-screen operations
Allows a procedure to be executed from any wait state
Sets the left margin of a printer
Adjusts the column width of a memo field output
DISPLAYS/does not display a cursor key menu during full
Allows a related message fot each PROMPT to be displayed
Sets up any open index file as the controlling file, where
Specifies path for file search
Sends/DOES NOT SEND output to a printer
Sets print output to LPT1,LPT2,COM1,COM2,file
Opens a named procedure file during compilation
Links two databases according to a key expression
PROMPTS/does not when a file is about to be overwritten
dBASE III/Clipper messages APPEAR/do not on the status
Halts/DOES NOT HALT execution after each command proc
SENDS/does not send the results of commands to the scree
PROMPTS/does not prompt for a catalog file title
Specifies the size of the type-ahead buffer
Opens a view (.vue) file
Determine number of lines in output at a specified line
Eliminate tab characters from text
Eliminate word-processing characters from text
Stores a character string in a memory variable
Computes the arithmetic mean of multiple expressions
Closes all database, index, format files & erases memvars
Erases current memory variables
DISPLAYS CURRENT MEMORY VARIABLES
Permits data entry to a GET field or variable
Erase current memory variables
Retrieves sets of saved memory variables
Copies current memory variables to a memory file
STORES EXPRESSIONS INTO MEMORY VARIABLES
ACCEPTS A SINGLE CHARACTER INTO A MEMORY VAR
MENU-DRIVEN

Sheet1

- Obtain an empty value of any data type expression
- Displays a design form to set up a label file (.LBL)
- Displays a design form to set up a report form file (.FRM)
- Creates a new screen file (.SCR)
- Identifies how the function keys are programmed
- Allows programmers to set the function keys to suit the

- DISPLAYS DATA IN THE FORM OF LABELS
- CREATES A COMMAND, FORMAT OR PROCEDURE FILE
- EDITS A LABEL FORM FILE
- MODIFIES A REPORT FORM FILE
- MODIFIES A SCREEN FILE
- Store a password so that it cannot be read
- Displays a tabular report of data
- Retrieve the current value of the specified SET parameter
- DBASE III PLUS VERSION NUMBER
- Provides an absolute value function
- COMPUTES AND DISPLAYS THE ARITHMETIC MEAN
- Convert a decimal number (base 10) to another number base
- Convert a different base value to decimal number (base 10)
- Convert a decimal number to binary (base 2)
- Obtain an integer greater than <expN>
- Convert a hexadecimal number (base 16) to a decimal
- Permits use of exponential functions
- Determine memory representation of <expN>
- Determine factorial of number
- Obtain an integer less than <expN>
- Convert a decimal number to hexadecimal (base 16)
- Provide an error return value for mathematical functions

LOGARITHM

- Give access to logarithms other than the natural logarithm
- Provide access to logarithms in the decimal system
- Determine memory representation of <expN>
- Locates the greater of two values

- Provides modulus math operations
- Raise a number to a power
- Compute cube roots, fifth roots, etc

- Totals fields with an active database file
- Ejects a page on the printer
- Determine if printer is ready to accept to accept output
- Switches output between two printers (Assumes LPT1 & LPT2)
- Determine printer status
- Send current screen image to print

Detect if printhead is positioned at top-of-form
Use anywhere in DO WHILE statement, nesting, recursion
Displays an expression list on the next line of the CRT
Draw and fill box with any CHR() to frame menus, draw boxes
Places menu selections on screen at the row,col location
Displays user-formatted data on the screen or printer
DRAWS A BOX
Displays/inputs data, prompts for input util expression .T.
CALL SEPARATELY COMPILED OR ASSEMBLED ROUTINE
Aborts program execution and returns to DOS control
CAUSES A PROGRAM OR PROCEDURE TO BE EXECUTED
Permits execution of several possible paths (CASES)
Permits structured loops in programs

NUMBER FOR ON ERROR CONDITION
Escapes a DO loop without terminating the program
DECLARE A SYMBOL FOR THE LINKER
Loops in optional increments or decrements
Declares user-defined functions
Allows conditional execution of commands in a program
Conditional processing of expressions. IIF() also supported

Stuffs keyboard buffer with given string then flushes buffer

LOOP returns control to the beginning of a DO WHILE...EN

Reads the contents of a text file into a memofield
To strip all the formatting characters from a memofield
Writes a character string to the specified disk file
On error message string
Specifies optional values that can be passed to a procedure
PRINTER COLUMN POSITION
Count parameters passed to a procedure
Hides memory variables from higher level command files
Marks the beginning of each routine in a procedure file

Makes memory variables global
Closes all files and exits from Clipper
FULL-SCREEN EXITING KEYPRESS

RETURNS TO CALLING PROGRAM AND REEXECUTES STATEMENT
Ends a program.
HALTS EXECUTION OF .PRG AND PRESENTS DOT PROMPT
Displays a block of text data in a command file

Sheet1

Draws a single or double line boxes or lines. Also CLEARs
Obtain the current screen attributes from the ROM BIOS
Erases or clears the CRT screen

Toggle the cursor on and off
Highlights an @..PROMPT and places prompt value in mem
Restores multiple screens saved with SAVE SCREEN TO <r
Read disk file directly into screen

Save current screen to disk file for later use
Obtain the current SET COLOR values
Scroll within a specified area on the screen
Sets screen display attributes

Provide private memory area to store data outside Clipper
Determine setting of BUFFERS in Config.sys
Executes a binary file (module) placed in memory with LOAD
Determine setting of COUNTRY in Config.sys

Release memory allocated with ALLOCATE()
Displays names of files on the specified disk drive
Provide access to INT 21H DOS function calls
Determine which version of PC-DOS or MS-DOS is running
Turns file attributes on or off
Read a text file from disk
Determine setting of FILES in Config.sys
Determine amount of disk space occupied by a file
Write a text file to disk
Attempts to lock a database(.dbt,.ntx in a shared enviroment
OPERATING SYSTEM ENVIRONMENT VARIABLES

Determine setting of LASTDRIVE in Config.sys
Places a binary file into memory where it can be executed
Used to attempt to lock a database(.dbt,.ntx in a shared

Detects network errors
Works with IBM PC Local Area Network
Determine the offset portion of a variable's memory address
Determine the location of a variable in memory
Determine the name of the operating system
Read data stored at specified memory locations
WRITE HEX BYTE TO MEMORY ADDRESS
Examine contents of CPU registers
Attempts to lock current database record. If the record was
Provide access to ROM BIOS function calls
Determine the segment portion of a variable's memory addre
Determine locatiob of a variable in memory
Determines the way in which database and related memo

Sheet1

Release file/record lock, UNLOCK ALL releases ALL locks
EXCLUSIVE opens file for exclusive(non-shared) use
Convert days to seconds
Convert hours to seconds
Determine if the time string is valid
Make an invalid time string valid
Convert minutes to seconds

Convert seconds to days
Convert seconds to hours
Convert seconds to minutes
Convert seconds to a time string
Convert time string to hours
Convert time string to minutes
Convert time string to seconds
Duplicates structure of file in USE to a new database

Displays an expression list on the current line of the CRT
Compares character types for a perfect match
The "!" may be used in place of the logical .NOT.
Allows comments on same line as command
Search for similar sounding names that are spelled

PURPOSE1,C,60

type of an array element.

.rray

processing or writing to a disk file
or writing to a disk file

a C array

current value

balance method
method

a specified amount

rest

a string

occur in <string1> regardless of the order
remove a character completely from a string

function

reconstruct the entire string

length

Displays 17 records per screen
without a qualifier

le

currently selected workarea

ILES

ILE

to close and reopen the files

3
\SE FILE

be validated and converted to a date type

parameter

ates

debugging or analysis
scrolls without pausing at each full-screen display

in HISTORY as they execute, default = OFF
ORY

in installation programs

in backup and installation programs
IN USE

with both types. Default = system monitor

the printer during execution, default = OFF

With SET ESCAPE OFF, the ESC is disabled

parison

more files

If SET FIELDS is ON, only fields in SET FIELDS TO are seen

:d

urs

Default width is 50

screen commands

message will appear on line <expN>

<n> is the number of the file in the index series

line

essed

n

length

IABLE

terminal in use

e

number (base 10)

f2)

S

IVES [WITH WORD()]

ED

program structure

NAME COMMAND

MPT

an area of the screen

```
var  
nemvar>
```

)

so two users cannot access a database at the same time

with the CALL command
environment)so two users cannot access database at same time

locked by same user, RLOCK() will release the lock

!SS

and index files are opened. The default setting is ON.

differently

```
EXAMPLE1,C,60
FOR i = pos TO LEN(array)-1 && equivalent
DECLARE textfiles[ADIR("*.TXT")] && declare array size
FOR i = start TO start + count - 1 && Equivalent
DO WHILE i > pos && equivalent ..where i = LEN(array)
```

```
* Sort array1 to array2
FOR i = start TO start + count -1 && Equivalent
```

```
DECLARE up_alpha[26]
```

```
ATLAST("a","AAABBaCdda") && Returns 10
ATNEXT("A", "AAABBaCdda", 0) && Returns 0
```

```
* Column is zero, <width> is screen size
```

```
CHRFOUND("cAb", "AAABBaCdda") && Returns .F.
CHRSWAP("AAABBaCdda", "A") && Returns "BBaCdda"
```

```
REPLACE ALL Field WITH ENCRYPT( Field, "My Password" )
EXPAND("Library",2)
? ISALPHA("ABC-123")
? ISLOWER("aBC-234")
? ISUPPER("aBC-234")
```

? "Dear "+LEFT("John J. Smith",4)
? LEN("James Smith")

? LOWER("THIS IS A NICE DAY")
? STR(3.145,10,2)
@ 1,0 SAY REPLICATE("*",20)

? STUFF('abc',2,1,'xyz') && Retrurns 'axyzc'

USE invoice

USE File2

USE Employee

USE Employee

USE Employee
USE Employee

FOR i TO FCOUNT()

Newfield = FIELDNAME(7)

SEEK memvar

USE Customers INDEX Name,Serial

? LUPDATE()

USE accounts

SKIP ALIAS 4 is the same as

Return date Dates between

* Compute beginning of current week

* All December 10, 1986 and return "19861210"

* If today is 06/01/87

conception = DATE()

? DAY(CHRISTMAS)

* If today is Saturday June 6th

xmas = CTOD("12/25/87")

m_drive = SPACE(1)

? DISKSPACE()

IF SELECT() =1

ON = <Esc> from GET...READ ignores VALID and Alt-C
*** INTERRUPTED ***

CLIPPER does not clear the screen before executing format

A wait state is defined as any command that pauses program-

USE Employee INDEX Lastname.ndx,Payrank.ndx,Areacode.nd
SET PATH TO C:\DBDATE\SALES

SET PRINTER TO LPT1

ACCEPT "Enter your last name...." TO LASTNAME && or

CREATE LABEL Maillist
CREATE REPORT Weekly
USE Employee

? HARDCR(notes)

CEILING(2.1) && Returns 3

? EXP(1.00000000)
? EXPONENT(100)
m_number = 13

acct_no = SPACE(6)

* To add "am" or "pm" to Time

? "Copyright 1987 EMULSIFIED SOFTWARE"

* To return to main menu from three levels deep

READ

* edit current memo

* Notes is a memofield

REPLACE Notes WITH MEMOTRAN(Notes," "," ")

status = MEMOWRIT("Temp.txt",Notes)

PARAMETER file

READ

```
? COL()
@ ....PROMPT "...
@ 8,10 PROMPT "ADD"
```

```
scr_value = SCRATTR()
* Clear all but screen border area
```

```
? TRANSFORM( 123456789, "###,###,###")
m_buffer = ALLOCATE( 1024 )
m_buffers = BUFFERS()
```

```
CALL DOSFUNC WITH m_regs,m_flags
IF VAL( DOSVER() ) < 3.1
```

```
Command      Cause of Failure
USE Netname INDEX By_name
```

```
? OS()
PEEKBYTE() 1-byte hex  PEEKDBL() 8-byte FP num
```

ON = Non-shared files

USE Accounts EXCLUSIVE

minutes = 285.61
start = SECONDS()

INDEX ON DESCEND(sales_date) TO date_dwn

```
EXAMPLE2,C,60
  array[i] = array[i +1]
ADIR("*.TXT", textfiles)    && fill array with names
  array[i] = val
  array[i] = array[i-1]
```

```
IF .NOT. ARRSORT("array1","array2")
  IIF array[i] = val,RETURN(1)
```

```
FOR i = 1 TO 26
```

```
ATLAST("b","AAABBaCdda") && Returns 0
ATNEXT("A", "AAABBaCdda", 1) && Returns 1
```

```
@ ROW(),0 SAY CENTER("Hello, world!",80)
```

```
CHRFOUND("CaB", "AAABBaCdda" ) && Returns .T.
CHRSWAP("AAABBaCdda","A","Z") && Returns "ZZZBBaCdd"
```

```
@...SAY DECRYPT( Field, "My Password" )
"L i b r a r y"
.T.
.T.
.F.
```


Dear John
11

this is a nice day
3.15

? STUFF('abc',2,1,") && Returns 'ac'

SELECT 2

APPEND FROM File1

CHANGE FIELDS Annual_pay && or

COUNT TO Nrecords

CREATE QUERY Findempl.qry
CREATE VIEW sample.vue

? FIELDNAME[i]

? Newfield

IF .NOT. FOUND()

SET ORDER TO 2

? "RECORDS IN FILE = ",LASTREC()

06/27/86

SELECT 2

SELECT 4

01/01/yy 01/01 - 03/31
m_weekbeg = BOW(DATE())

CDTOS("121086", "MMDDYY")

? CMONTH(DATE())

add_months = 9

25

? DOW(DATE())

? DTOS(xmas) &&Returns "19871225"

@ 12,10 SAY "Enter drive letter: ";

309248

@ 23,0 SAY "Editing main file"

allows for program termination.
Called from <filename.prg>

files. However commands are allowed in format files. When

WAIT, READ, ACCEPT, INPUT, MENU TO

SET ORDER TO 3

SET PRINTER TO Accounts.prn

ACCEPT TO LASTNAME

CREATE SCREEN Showempl.scr

CEILING(.3) && Returns 1

2.71828183

6

? FACT(m_number)

@ 1,0 SAY "Enter account" GET acct_no

? TIME() + IF(TIME() < "12","am","pm")

? INKEY(0) && Halt execution

last_key = LASTKEY()

REPLACE memo_field WITH MEMOEDIT(memo_field,5,10,20

REPLACE Notes WITH MEMOREAD("Temp.txt")

IF status

IF PCOUNT() =0

IF UPDATED()

```
5  
CALL CURSOR WITH "off"  
@ 9,10 PROMPT "EDIT"
```

```
<code that changes the screen>  
CALL SCROLL WITH 1,1,22,78,0,"U"
```

```
* Returns 123,456,789  
IF "" = m_buffer && null means an error  
IF m_buffers = -1
```

```
IF m_flags %2 = 1  
? "This software requires DOS 3.1 or later."
```

```
USE USE...EXCLUSIVE by another process  
SEEK NETNAME()
```

```
DOS 3.10  
PEEKCHAR() 1-byte ascii PEEKSTR() Nul-terminated ascii
```

OFF = Shared files

```
IF .NOT. NETERR()
```

```
secs = MTOSS( minutes )  
DO process
```


EXAMPLE3,C,60
NEXT

NEXT
i = i - 1

? "Error sorting array"
NEXT

up_alpha[i] = CHR(64 +i)

ATLAST("A","AAABBaCdda") && Returns 3
ATNEXT("A", "AAABBaCdda", 3) && Returns 3

CHRFOUND("b", "AAABBaCdda") && Returns .F.
CHRSWAP("AAABBaCdda","a","z") && Returns "AAABBzdd"

? ISUPPER("Abc-234")

? LTRIM(STR(3.145,10,2))

? STUFF('abc',2,0,'xyz') && Returns 'axyzbc'

USE client

COUNT FOR Annual_pay>="50000" .AND. Male TO Richmei

NEXT

? "Record not in file"

? INDEXKEY(1) && Returns Name.ntx key expression

USE temp

SKIP

04/01/yy 04/01 - 06/30

CDTOS("86/12/10", "YY/MM/DD")

June

due_month = ((MONTH(conception)-1 + add_month) %12) +

6

GET m_drive VALID ISDRIVE(m_drive)

ELSE

what now - Cancel, Ignore, Suspend?(C,I, or S)

use .CLP files with a program that has SET FORMAT TO

CEILING(-0.3) && Returns 0

IF EMPTY(acct_no)

CLEAR && til key press then clear screen
KEYBOARD "Q"+CHR(13)+"Q"+CHR(13)+"Q"+CHR(13)
IF last_key <> 27

* display current memo

RUN Editor Temp.txt

ACCEPT "File to use:" TO file

REPLACE ID WITH m_ID

MENU TO foo
@ 10,10 PROMPT "Maintenance"

- * Restore original screen
- * Scroll box in middle of screen 3 lines down

<error message>
? "ERROR --- cannot find Config.sys"

? "Invalid country code"
QUIT

USE..EXCLUSIVE USE...EXCLUSIVE by another process
IF FOUND()

PEEKINT() 2-byte int

SET INDEX TO Acct_ID

elapsed = SECONDS() - Start

SEEK DESCEND(find_date)

EXAMPLE4,C,60

ENDDO

ENDIF

NEXT

ATNEXT("a", "AAABBaCdda", 1) && Returns 6

CHRFOUND("dddd", "AAABBaCdda") && Returns .T.
z"

? ISALPHA("123-ABC")
? ISLOWER("Abc-234")
.T.

3.15

? ALIAS(1),ALIAS() && Returns "invoice client"

USE Employee

n

ENDIF

? INDEXKEY(0) && Returns Serial.ntx key expression

SELECT accounts

SELECT 1

07/01/yy 07/01 - 09/30

CDTOS("10.12.1986", "DD.MM.YYYY")

? "Your baby is due in the month of " + CMONTHD(due_month)

READ

@ 23,0 SAY "Editing customer file"

OFF = Will not allow an escape to terminate a READ. In

statements, the .FMT files must have .PRG file extensions.

Note that INKEY() is not a wait state

CEILING(-2.9) && Returns -2

RETURN

REPLACE memo_field WITH MEMOEDIT(memo_field,5,10,20,69,.F.)

ENDIF

ENDIF

ENDIF


```
CALL CURSOR WITH "on"  
MENU TO choice
```

```
SET COLOR TO &scr_value  
CALL SCROLL WITH 10,30,14,50,3,"d"
```

```
RETURN    && abort this routine  
IF m_buffers < 10 .OR. m_buffers > 15
```

```
ENDIF  
ENDIF
```

```
APPEND BLANK  FLOCK() by another process or two  
? "Welcome" +NETNAME()
```

```
PEEKLONG() 4-byte int
```

ENDIF

? "Process took "+ STR(Elapsed) + " seconds"

Sheet1

	TR.LIB	Yes	RETTIG
		Yes	PLUS
		Yes	Yes PLUS
	TR.LIB	Yes	RETTIG
		Yes	PLUS
		Yes	PLUS
	CLIPPER.LIB		NANTUCKET PLUS
	TR.LIB	Yes	RETTIG
	CLIPPER.IIB		NANTUCKET
	CLIPPER.LIB		NANTUCKET
		Yes	PLUS
	CLIPPER.LIB		NANTUCKET
	CLIPPER.LIB		NANTUCKET
	CLIPPER.LIB		NANTUCKET
	CLIPPER.LIB		NANTUCKET
	TR.LIB	Yes	RETTIG
	CLIPPER.LIB		NANTUCKET
	CLIPPER.LIB		NANTUCKET
	CLIPPER.LIB		NANTUCKET
		Yes	Yes PLUS
			CLIP
	CLIPPER.LIB	Yes	Yes NANTUCKET
		Yes	PLUS
CHANGE FIELDS Area_code,Phone_no FOR Area_code="206"		Yes	PLUS
	CLIPPER.LIB		NANTUCKET
	CLIPPER.LIB		NANTUCKET
	CLIPPER.LIB		NANTUCKET
	CLIPPER.LIB	Yes	Yes NANTUCKET
	CLIPPER.LIB		NANTUCKET
	CLIPPER.LIB		NANTUCKET
		Yes	PLUS
		Yes	PLUS
	CLIPPER.LIB	Yes	Yes NANTUCKET
	CLIPPER.LIB		Yes NANTUCKET
	CLIPPER.LIB		NANTUCKET
		Yes	No PLUS
		Yes	dBASE PLUS
		Yes	Yes NANTUCKET/AT
	CLIPPER.LIB		NANTUCKET PLUS
	CLIPPER.LIB	Yes	NANTUCKET PLUS
	CLIPPER.LIB		NANTUCKET
	CLIPPER.LIB	Yes	Yes NANTUCKET
	TR.LIB		Yes RETTIG
	CLIPPER.LIB		Yes NANTUCKET
	TR.LIB	Yes	RETTIG
	CLIPPER.LIB		NANTUCKET

Sheet1

TR.LIB	Yes		RETTIG PLUS
CLIPPER.LIB			NANTUCKET
CLIPPER			NANTUCKET
CLIPPER.LIB	Yes	Yes	NANTUCKET
CLIPPER.LIB	Yes	Yes	NANTUCKET
CLIPPER.LIB		Yes	NANTUCKET
CLIPPER.LIB	Yes	Yes	NANTUCKET
CLIPPER.LIB	Yes	Yes	NANTUCKET
	Yes		PLUS PLUS PLUS PLUS
	Yes	No	PLUS
TR.LIB			RETTIG
TR.LIB	Yes		RETTIG
TR.LIB	Yes		RETTIG
CLIPPER.LIB	Yes	Yes	NANTUCKET
CLIPPER.LIB	Yes	Yes	NANTUCKET
CLIPPER.LIB	Yes	Yes	NANTUCKET
			PLUS PLUS
CLIPPER.LIB	Yes	Yes	NANTUCKET
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB			NANTUCKET
CLIPPER			NANTUCKET
	Yes		PLUS
CLIPPER.LIB	Yes	Yes	NANTUCKET/AT
CLIPPER.LIB		Yes	NANTUCKET
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB			NANTUCKET
CLIPPER			NANTUCKET
			PLUS PLUS
TR.LIB	Yes		RETTIG
CLIPPER.LIB			NANTUCKET
			PLUS PLUS
CLIPPER.LIB			NANTUCKET
TR.LIB	Yes		RETTIG
TR.LIB	Yes		RETTIG
CLIPPER.LIB			NANTUCKET
TR.LIB	Yes		RETTIG
CLIPPER.LIB	Yes	Yes	NANTUCKET
TR.LIB		Yes	RETTIG
CLIPPER.LIB	Yes	Yes	NANTUCKET
CLIPPER.LIB	Yes	Yes	NANTUCKET
CLIPPER		Yes	NANTUCKET

10/01/yy 10/01 - 12/31

addition, the Alt-C is ignored.

CLIPPER.LIB			NANTUCKET
	Yes		AT
CLIPPER.LIB	No	Yes	NANTUCKET
	Yes	No	AT
CLIPPER.LIB			NANTUCKET
	Yes	No	AT
	Yes	No	AT
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB	No	Yes	NANTUCKET
	Yes		AT
CLIPPER.LIB		Yes	NANTUCKET
	Yes		AT
	Yes		AT
CLIPPER.LIB		Yes	NANTUCKET
CLIPPER.LIB		Yes	NANTUCKET
CLIPPER.LIB	Yes	Yes	NANTUCKET
	Yes		PLUS
	Yes		AT
CLIPPER.LIB		Yes	NANTUCKET
	Yes		PLUS
CLIPPER.LIB	Yes	Yes	NANTUCKET
CLIPPER.LIB	Yes	Yes	NANTUCKET
CLIPPER		Yes	NANTUCKET
CLIPPER.LIB	Yes	Yes	NANTUCKET
CLIPPER.LIB	Yes	Yes	NANTUCKET
	Yes		AT
CLIPPER.LIB	Yes	Yes	NANTUCKET/AT
	Yes		AT
	Yes	No	AT
	Yes	No	AT
	Yes		AT
	Yes		AT
TR.LIB		Yes	RETTIG
TR.LIB		Yes	RETTIG
TR.LIB		Yes	RETTIG
CLIPPER.LIB	Yes	Yes	NANTUCKET/AT
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB			NANTUCKET
	Yes	No	PLUS
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB	Yes	Yes	NANTUCKET
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB			NANTUCKET
			PLUS
			PLUS
	Yes		PLUS

Sheet1

TR.LIB	Yes	RETTIG
	Yes	PLUS
	Yes	PLUS
	Yes	PLUS
	Yes No	AT
	Yes	AT
MEMO.LIB	Yes	NANTUCKET
	Yes	PLUS
	Yes No	PLUS
		PLUS
		PLUS
		PLUS
TR.LIB	Yes	RETTIG
CLIPPER.LIB		NANTUCKET
TR.LIB	Yes	RETTIG
		PLUS
CLIPPER.LIB		NANTUCKET
		PLUS
TR.LIB		RETTIG
TR.LIB		RETTIG
TR.LIB		RETTIG
TR.LIB	Yes	RETTIG
TR.LIB	Yes	RETTIG
	Yes	PLUS
TR.LIB		RETTIG
TR.LIB	Yes	RETTIG
TR.LIB	Yes	RETTIG
TR.LIB	Yes	RETTIG
TR.LIB	Yes	RETTIG
TR.LIB	Yes	RETTIG
CLIPPER.LIB	Yes	NANTUCKET
	Yes	PLUS
TR.LIB	Yes	RETTIG
TR.LIB	Yes	RETTIG
TR.LIB	Yes	RETTIG
	Yes	PLUS
CLIPPER.LIB	Yes	NANTUCKET
	Yes	PLUS
TR.LIB	Yes	RETTIG
TR.LIB	Yes	RETTIG
CLIPPER.LIB		NANTUCKET
CLIPPER.LIB		NANTUCKET
CLIPPER.LIB	Yes Yes	NANTUCKET/AT
CLIPPER.LIB		NANTUCKET
CLIPPER.LIB	Yes Yes	NANTUCKET/AT
TR.LIB	Yes	RETTIG
TR.LIB		RETTIG
TR.LIB	Yes	RETTIG
TR.LIB		RETTIG
CLIPPER.LIB		NANTUCKET

Sheet1

TR.LIB	Yes		RETTIG
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB			NANTUCKET PLUS
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB	Yes	Yes	NANTUCKET/AT
CLIPPER.LIB	Yes	Yes	NANTUCKET
CLIPPER.LIB	Yes	Yes	NANTUCKET
CLIPPER		Yes	NANTUCKET PLUS
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB		Yes	NANTUCKET
CLIPPER.LIB		Yes	NANTUCKET
CLIPPER.LIB		Yes	NANTUCKET
CLIPPER.LIB	Yes	Yes	NANTUCKET
CLIPPER.LIB		Yes	NANTUCKET
CLIPPER.LIB			NANTUCKET
CLIPPER	Yes	Yes	NANTUCKET
CLIPPER		Yes	NANTUCKET
CLIPPER.LIB		Yes	NANTUCKET
CLIPPER.LIB	Yes	Yes	NANTUCKET
MEMO		Yes	NANTUCKET
MEMO.LIB		Yes	NANTUCKET
MEMO.LIB		Yes	NANTUCKET
MEMO.LIB		Yes	NANTUCKET
	Yes		PLUS
CLIPPER			NANTUCKET PLUS
CLIPPER.LIB		Yes	NANTUCKET
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB	Yes	Yes	NANTUCKET
CLIPPER.LIB		Yes	NANTUCKET
CLIPPER.LIB		Yes	NANTUCKET
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB	Yes	Yes	NANTUCKET
CLIPPER.LIB			NANTUCKET PLUS
CLIPPER.LIB			NANTUCKET PLUS
CLIPPER.LIB			NANTUCKET PLUS
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB			NANTUCKET
CLIPPER.LIB			NANTUCKET

USE &file

	CLIPPER.LIB	Yes	NANTUCKET
	TR.LIB	Yes	RETTIG
	CLIPPER.LIB		NANTUCKET
	CLIPPER.LIB		NANTUCKET
	TR.LIB		RETTIG
	CLIPPER	Yes	NANTUCKET
			CLIP
	TR.LIB		RETTIG
	CLIPPER.LIB		NANTUCKET
	TR.LIB		RETTIG
	TR.LIB		RETTIG
	TR.LIB		RETTIG
	CLIPPER.LIB	Yes	NANTUCKET
	CLIPPER.LIB		NANTUCKET
	TR.LIB		RETTIG
? "ERROR --- set BUFFERS between 10-15 in Config.sys	TR.LIB	Yes	RETTIG
		Yes	PLUS
	TR.LIB	Y	RETTIG
	CLIPPER.LIB		NANTUCKET
	TR.LIB	Yes	RETTIG
	CLIPPER.LIB		NANTUCKET
	TR.LIB	Yes	RETTIG
	TR.LIB	Yes	RETTIG
	TR.LIB	Yes	RETTIG
	TR.LIB	Yes	RETTIG
	TR.LIB	Yes	RETTIG
	TR.LIB	Yes	RETTIG
			RETTIG
	CLIPPER.LIB	Yes	NANTUCKET
			PLUS
	CLIPPER.LIB	Yes	NANTUCKET
	TR.LIB	Yes	RETTIG
		Yes	No PLUS
	CLIPPER.LIB	Yes	NANTUCKET
	CLIPPER.LIB	Yes	NANTUCKET
APPEND BLANK attempt at the same time	CLIPPER.LIB		NANTUCKET
ENDIF	CLIPPER.LIB	Yes	NANTUCKET
	TR.LIB	Yes	RETTIG
	TR.LIB	Yes	RETTIG
		Yes	PLUS
	TR.LIB	Yes	RETTIG
	TR.LIB	Yes	RETTIG
	TR.LIB	Yes	RETTIG
	CLIPPER.LIB	Yes	NANTUCKET
	TR.LIB	Yes	RETTIG
	TR.LIB	Yes	RETTIG
			RETTIG
	CLIPPER.LIB	Yes	NANTUCKET
	CLIPPER.LIB		NANTUCKET

Sheet1

CLIPPER	Yes	NANTUCKET
CLIPPER		NANTUCKET
TR.LIB	Yes	RETTIG
TR.LIB	Yes	RETTIG
TR.LIB	Yes	RETTIG
TR.LIB	Yes	RETTIG
TR.LIB	Yes	RETTIG
CLIPPER.LIB		NANTUCKET
TR.LIB	Yes	RETTIG
TR.LIB	Yes	RETTIG
TR.LIB	Yes	RETTIG
TR.LIB	Yes	RETTIG
TR.LIB	Yes	RETTIG
TR.LIB	Yes	RETTIG
TR.LIB	Yes	RETTIG
CLIPPER.LIB		NANTUCKET
EXTRA.LIB		
CLIPPER.LIB		NANTUCKET
CLIPPER.LIB		NANTUCKET
CLIPPER.LIB		NANTUCKET
CLIPPER.LIB		NANTUCKET
TR.LIB	Yes	RETTIG