PRAGMA REFERENCE MANUAL

$INPUT $N Standard Verb Input/Output Operation
Accepts numerical input with two decimal places (such as dollars and cents)
from the keyboard and writes it in a noun you designate.
You may include a leading minus sign, with a maximum length of 10, including
2 decimal places and a point. A leading plus sign is not possible.
The decimal point is not mandatory.
LENGTH contains a count of the characters accepted.
$INPUT into the noun NOUN

NOUN is the noun you want the input data to be copied to.

SEE ALSO: $INPUT LAYOUT/INPUT/INPUT NUMBER
DISCUSSION

The only data acceptable to $INPUT is a number with two decimal places. You may only include a leading minus sign. The plus sign is only accepted if a leading minus is present, else it beeps. A decimal point is not mandatory.

$INPUT basically acts like an INPUT NUMBER with a layout of 10.2. The same editing keys as in INPUT NUMBER are allowed.

Every $INPUT operation (as well as every other INPUT, CUT, JOIN, or RECEIVE) writes a new value to the target noun LENGTH. To preserve the contents of LENGTH, copy it to another noun before using $INPUT, INPUT, CUT, JOIN, or RECEIVE again.

EXAMPLE

$INPUT into the noun CURRENT PRICE

Data is accepted and written into the noun CURRENT PRICE.

~~~~~~~~~~~~~~~~~~~~~~~~

DOS CLOSE READ DCR Standard Verb Operating System Operation
Closes a text file after having opened it with DOS OPEN READ and read from it
with DOS READ or DOS READ LAYOUT.
You cannot specify a filename because you can open only one file at a time.

The noun IPC STATUS contains the result of the DOS CLOSE READ operation.

DOS CLOSE READ
~~~~~~~~~~~~~~~~~~~~~~~~

SEE ALSO: DOS OPEN READ/DOS READ/DOS READ LAYOUT
~~~~~~~~~~~~~~~~~~~~~~~~~~

DOS CLOSE WRITE   DCW Standard Verb Operating System Operation
Closes a file after having written to it. You cannot specify a filename in this verb because you can only write to one file at a time.

The noun IPC STATUS contains the result of the DOS CLOSE WRITE operation.

DOS CLOSE WRITE

SEE ALSO: DOS CLOSE READ/DOS WRITE/
~~~~~~~~~~~~~~~~~~~~~~~~~

DOS COPY   DC   Standard Verb Operating System Operation
Copies an operating system file to another operating system file.

The noun IPC STATUS contains the result of the DOS COPY operation.

DOS COPY object is FILENAME

FILENAME is an expression or a noun containing the SOURCEFILE and the DESTINATIONFILE. SOURCEFILE and DESTINATIONFILE must be separated by a space.

SEE ALSO: DOS EXIST/IPC STATUS/
DOS COPY (DC)

DOS COPY copies an operating system file to another operating system file. The result of the operation will be written to the standard noun IPC STATUS (IS). If the file you want to copy exists, the noun IPC STATUS will contain OK and the file will be copied to the specified destination file. If the file you want to copy does not exist, the IPC STATUS will contain FILE NOT FOUND.

If the destination file does not exist, it will be created.

DOS COPY lets you copy a file in any directory or drive to a file in any other directory or drive by simply preceding the filenames with the drive and path. If the destination directory does not exist, the IPC STATUS will contain FILE NOT FOUND.

DOS COPY also lets you copy a file to the console or to the printer, in which case the contents of the file (which must be a textfile) will either be displayed on the screen or printed by the printer. To copy to the console write CON in place of the name of the destination file. To copy to the printer write PRN in place of the destination file.

EXAMPLES

VERB called TEST DOS COPY

```
1  DOS COPY object is "TEST.PFM ¥BACKUPS¥TEST.PFM"
2  DISPLAY the value IPC STATUS
```

This example copies the file TEST.PFM that resides in the current directory to a file with the same name but in the directory ¥BACKUPS.

The name of the sourcefile and the name of the destination file must always be separated by a space.

VERB called TEST DOS COPY TO CONSOLE

```
1  DOS COPY object is "C:¥AUTOEXEC.BAT CON"
2  IF the value IPC STATUS "<>" the value "OK
       do
          DISPLAY the value IPC STATUS
       else
       end
3  INPUT KEY into the noun KEY
```

This example copies to the console (i.e., displays) the file AUTOEXEC.BAT which resides in the root directory of your C drive. In the unlikely event that you don't have an AUTOEXEC.BAT file the verb will display the IPC STATUS as FILE NOT FOUND.

To copy the file AUTOEXEC.BAT to your printer (i.e., print) you would substitute the expression "C:¥AUTOEXEC.BAT CON" with the expression "C:¥AUTOEXEC.BAT PRN"

The last line with the verb INPUT KEY is only there to stop the program after the display.

It is important that you always check the IPC STATUS since you are performing an operation on the operating system level where things are beyond the control of PRAGMA.

Please take note that the examples shown above are for the DOS operating system.

~~~~~~~~~~~~~~~~~~~~~~~~

DOS DELETE   DD   Standard Verb Operating System Operation
Deletes an operating system file.

The noun IPC STATUS contains the result of the DOS DELETE operation.

DOS DELETE filename FILENAME

FILENAME is an expression or a noun containing the name and optional path of
the operating system file that has to be deleted.

SEE ALSO: IPC STATUS/ /
DOS DELETE (DD)

DOS DELETE deletes an operating system file. The result of the operation will be written to the standard noun IPC STATUS (IS). If the file you want to delete exists, the noun IPC STATUS will contain OK and the file will have been deleted. If the file does not exist, the IPC STATUS will contain FILE NOT FOUND.

DOS DELETE lets you delete a file in any directory or drive you wish by simply preceding the filename with the drive and path of the file.

EXAMPLE

VERB called TEST DOS DELETE

1 COPY the value "TESTFILE.TXT" to the noun FILENAME
2 DOS DELETE filename FILENAME
3 DISPLAY the value IPC STATUS

This example deletes the file TESTFILE.TXT in the current directory.

It is important to always check the IPC STATUS since you are performing an operation on the operating system level where things are beyond the control of PRAGMA.

~~~~~~~~~~~~~~~~~~~~~~~~

DOS EXIST   DE   Standard Verb Operating System Operation
Checks the existence of an operating system file.

Returns OK in the noun IPC STATUS if the operating system file specified
exists and FILE NOT FOUND if the file is not there.

DOS EXIST filename FILENAME

FILENAME is an expression or a noun containing the name (and path) of the
operating system file.

SEE ALSO: IPC STATUS/ /
DOS EXIST (DE)

DOS EXIST looks for the existence of an operating system file. The result of the operation is written to the standard noun IPC STATUS (IS). If the file you are testing for exists, the noun IPC STATUS will contain OK. If the file does not exist, the IPC STATUS will contain FILE NOT FOUND.

DOS EXIST lets you check for a file in any directory or drive you wish by simply preceding the filename with the drive and path of the file.

EXAMPLE

VERB called TEST DOS EXIST

```
1  DOS EXIST filename "C:¥AUTOEXEC.BAT"
2  DISPLAY the value IPC STATUS
```

This example looks for the existence of the file AUTOEXEC.BAT in the root directory of drive C.

It is important to always check the IPC STATUS since you are performing an operation on the operating system level where things are beyond the control of PRAGMA.

~~~~~~~~~~~~~~~~~~~~~~~~

DOS OPEN APPEND DOA Standard Verb Operating System Operation
Opens the specified file for writing in append mode. If the file does not exist it is created. This verb is always used with the verb DOS WRITE. After having written to the file with DOS WRITE you must close it with the verb DOS CLOSE WRITE. If you want to write to a new file you must use the verb DOS OPEN CREATE.
The noun IPC STATUS contains the result of the operation.
DOS OPEN APPEND filename FILENAME
DOS WRITE the value TEXT
~~~~~~~~~~~~~~~~~~~~~~~~

DOS CLOSE WRITE

FILENAME is an expression or noun containing the name (and path) of the oper-
ating system file you want to append the noun or expression TEXT.
SEE ALSO: DOS OPEN CREATE/DOS WRITE/DOS CLOSE WRITE
~~~~~~~~~~~~~~~~~~~~~~~~~

DOS OPEN CREATE   DOC Standard Verb Operating System Operation
Creates an empty operating system file and opens it for writing. If you want
to write to the file you must use the verb DOS WRITE. After having written
to the file you must close it with the verb DOS CLOSE WRITE. If you want to
append to an existing file you must use the verb DOS OPEN APPEND.

The noun IPC STATUS contains the result of the operation.
DOS OPEN CREATE filename FILENAME
DOS WRITE the value TEXT
DOS CLOSE WRITE

FILENAME is an expresion or noun containing the name (and path) of the oper-
ating system file you want to create.
SEE ALSO: DOS OPEN APPEND/DOS WRITE/DOS CLOSE WRITE
~~~~~~~~~~~~~~~~~~~~~~~~~~

DOS OPEN READ   DOR Standard Verb Operating System Operation
Opens an operating system text file so that you may then do a DOS READ or a
DOS READ LAYOUT and read the contents of that file into a noun.
After doing a DOS READ or a DOS READ LAYOUT always run the verb DOS CLOSE
READ.
The noun IPC STATUS contains the result of the operation DOS OPEN READ.

DOS OPEN READ filename FILENAME
DOS READ into the noun TEXT
DOS CLOSE READ

FILENAME is an expression or noun containing the name (and path) of the opera-
ting system file you want to read into the noun TEXT.
SEE ALSO: DOS READ/DOS READ LAYOUT/DOS CLOSE READ
~~~~~~~~~~~~~~~~~~~~~~~~~

DOS READ   DR   Standard Verb Operating System Operation
After having opened with DOS OPEN READ the text file that you want to read
from, you can read the contents of the file with the verb DOS READ.
The contents of the file will be put into the noun that is the object.
A maximum of 2 kilobytes can be read at a time. The RECEIVE TERM CHAR will
terminate reading the file. After DOS READ you must run DOS CLOSE READ.
IPC STATUS contains the result, LENGTH the number of characters received.
DOS OPEN READ filename FILENAME
DOS READ into the noun TEXT
DOS CLOSE READ

FILENAME is an expression or noun containing the name (and path) of the opera-
ting system file you want to read into the noun TEXT.
~~~~~~~~~~~~~~~~~~~~~~~~~

SEE ALSO: DOS OPEN READ/DOS CLOSE READ/DOS READ LAYOUT
DOS OPEN READ (DOR)
DOS READ (DR)
DOS READ LAYOUT (DRL)
DOS CLOSE READ (DCR)

DOS OPEN READ opens an operating system text file so that a DOS READ or a DOS READ LAYOUT reads all or part of that file into a noun. Only text files can be accessed, binary files not. You can open only one file at a time.

DOS READ terminates reading the file when it encounters the RECEIVE TERM CHAR specified in PRAGMA.DES, when it encounters a null (_@) or when it has read 2048 characters.

DOS READ LAYOUT terminates reading the file when it has read the number of characters specified in the layout number, when it encounters the RECEIVE TERM CHAR specified in PRAGMA.DES or if it encounters a null (_@). The layouts of DOS READ LAYOUT can be greater than 2048.

A null always terminates a read because null is a string terminator in PRAGMA (and the C language).

When DOS READ or DOS READ LAYOUT terminate because of the receive terminating character, the receive terminating character is always removed from the input data.

DOS READ and DOS READ LAYOUT will put the number of characters read into the standard noun LENGTH (LT).

After having finished reading, the operating system file must always be closed by running DOS CLOSE READ.

Every one of these verbs will write the results of the operation whether it was successful or not into the standard noun IPC STATUS (IS).

If you try to read a file and there is no more data to be read, you will get an END OF FILE message in the IPC STATUS. If any data is available, even just one byte, you will get an OK message. This allows you to read a large text file by performing various reads, as shown in the second example below.

For a list of all the possible IPC STATUS messages consult the IPC STATUS reference.

EXAMPLES

VERB called TEST DOS READ

```
1  DOS OPEN READ filename "TEST.TXT"
2  IF the value IPC STATUS "<>" the value "OK"
       do
3         GO TO line labeled ERROR
       else
```

```
      end
4  DOS READ into the noun TEXT
5  IF the value IPC STATUS "<>" the value "OK"
      do
6        GO TO line labeled ERROR
      else
      end
7  DOS CLOSE READ
8  IF the value IPC STATUS "<>" the value "OK"
      do
9        GO TO line labeled ERROR
      else
      end
10  MOVE CURSOR DOWN the amount 1
11  DISPLAY the value TEXT
12  MOVE CURSOR DOWN the amount 1
13  DISPLAY the value LENGTH
14  RETURN
15  LABEL this line with
ERROR
16  DISPLAY the value IPC STATUS
```

It is important to always check the IPC STATUS since you are performing an operation on the operating system level where things are beyond the control of PRAGMA.

The verb reads a textfile TEST.TXT, which must be present in the same directory where PRAGMA is, since no path is indicated.

The whole file will be read, if the file has less than 2048 characters. If it is larger 2048 characters it will read up to 2048 characters unless it encounters a terminating character specified in RECEIVE TERM CHAR.

You can substitute line 4 with DOS READ LAYOUT, which lets you specify the number of characters that will be read. But remember that the terminating character specified in RECEIVE TERM CHAR takes precedence, and reading will be terminated upon finding that character, regardless of how many characters have been read. If you specify a number of characters larger than the file you want to read from, the IPC will contain an OK message, and the receiving noun will contain the complete file.

If you need to read a large file into a noun it is better to execute a DOS READ until the end of the file has been reached. The following example illustrates the principle:

VERB called TEST READ LONG FILE

```
1  COPY the value "" to the noun TEXT
2  DOS OPEN READ filename "TEST.TXT"
3  IF the value IPC STATUS "<>" the value "OK"
```

```
      do
4        GO TO line labeled ERROR
      else
      end
5  LABEL this line with
LOOP
6  DOS READ into the noun HEAD
7  JOIN the value TEXT to the value HEAD
8  COPY the value HEAD to the noun TEXT
9  IF the value IPC STATUS "<>" the value "OK"
      do
10        DOS CLOSE READ
      else
11        GO TO line labeled LOOP
      end
12  MOVE CURSOR DOWN the amount 1
13  DISPLAY the value TEXT
14  RETURN
15  LABEL this line with
ERROR
16  DISPLAY the value IS
```

In this verb DOS READ will read the file until it has read the maximum number of characters it can read and put the result into the standard noun HEAD. The contents of HEAD are then JOINed to the contents of the noun TEXT, thus always adding whatever is read to the contents of TEXT. When there is nothing left to be read the IPC STATUS contains END OF FILE rather than OK, the loop terminates, the file is closed and the program continues and the noun TEXT will be displayed.

If the receive terminating character is encountered DOS READ will stop reading, but the program will loop, and DOS READ will read again, until the receive terminating character is met again and so on, until the END OF FILE has been reached. The difference will be that the receive terminating character will have been eliminated from the contents of the noun TEXT, since the receive terminating character is always removed from the input data.

~~~~~~~~~~~~~~~~~~~~~~~~~

DOS RENAME DRN Standard Verb Operating System Operation
Renames an existing OS file or moves a file from one subdirectory to another.

The noun IPC STATUS contains the result of the operation DOS RENAME.

DOS RENAME object is FILENAME ("PATH¥OLDNAME PATH¥NEWNAME")

FILENAME is an expression or a noun containing the OLDNAME and the NEWNAME.
OLDNAME and NEWNAME can have a path. If the subdirectory is not the same, the file will be moved from one subdirectory to the other.

SEE ALSO: DOS COPY/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

DOS RENAME (DRN)

DOS RENAME renames an operating system file or moves a file from one directory to another.

The result of the operation will be written to the standard noun IPC STATUS (IS). If the file you want to copy exists, the noun IPC STATUS will contain OK and the file will be renamed or moved. If the file you want to rename does not exist, the IPC STATUS will contain FILE NOT FOUND.

EXAMPLES

VERB called TEST DOS RENAME

```
1  DOS COPY object is "TEST.TXT NEWTEST.TXT"
2  DISPLAY the value IPC STATUS
```

This example renames the file TEST.TXT that resides in the current directory to NEWTEST.TXT. If the file TEST.TXT does not exist the verb will display the IPC STATUS as FILE NOT FOUND.

The original name of the file and the new name of the file must always be separated by a space.

VERB called TEST DOS MOVE

```
1  DOS RENAME object is "TEST.TXT C:¥TEST.TXT"
2  DISPLAY the value IPC STATUS
```

This example moves the file TEST.TXT from the current directory to the root directory of your C drive.

It is important that you always check the IPC STATUS since you are performing an operation on the operating system level where things are beyond the control of PRAGMA.

Please take note that the examples shown above are for the DOS operating system.

~~~~~~~~~~~~~~~~~~~~~~~~~

DOS WRITE DW Standard Verb Operating System Operation
After having opened with DOS OPEN APPEND or DOS OPEN CREATE an operating system file, you can write to the file with the verb DOS WRITE.
After having finished writing to the file you must then close it with the verb DOS CLOSE WRITE.
~~~~~~~~~~~~~~~~~~~~~~~~~

The noun IPC STATUS contains the result of the operation DOS WRITE.
DOS OPEN APPEND filename FILENAME
DOS WRITE the value TEXT
DOS CLOSE WRITE
FILENAME is an expression or noun containing the name and optional path of the operating system file you want to write the noun or expression TEXT to.

SEE ALSO: DOS OPEN APPEND/DOS OPEN CREATE/DOS WRITE LAYOUT
DOS OPEN CREATE (DOC)
DOS OPEN APPEND (DOA)
DOS WRITE (DW)
DOS WRITE LAYOUT (DWL)
DOS CLOSE WRITE

DOS OPEN CREATE creates and opens an operating system text file so that a DOS WRITE or a DOS WRITE LAYOUT writes the contents of a noun to the text file. If a file with that name already exists, it will be overwritten with an empty file of the same name.

DOS OPEN APPEND opens the specified operating sytem text file so that a DOS WRITE or a DOS WRITE LAYOUT writes the contents of a noun to the text file. If the file already contains text, the new text will be appended to it. If a file with that name does not exist, it will be created.

You can open only one file at a time, either with DOS OPEN CREATE or with DOS OPEN APPEND.

DOS WRITE will write the whole contents of the noun, without a size limit to the opened file. DOS WRITE LAYOUT will write the number of characters specified in the layout.

DOS WRITE and DOS WRITE LAYOUT will also append at the end of every write the SEND TERM CHARACTER specified in PRAGMA.DES.

If SEND TERM CHARACTER is not specified in PRAGMA.DES, DOS WRITE and DOS WRITE LAYOUT will append the default character, which is a null character (_@). Be careful with the null character. The null character is a string terminator in PRAGMA (and the C language). A DOS READ will always stop when it encounters a null character. This may or may not be desirable, depending upon the format of the data in the text file being read at the time. The SEND TERM CHARACTER must be set correctly when writing the file to correspond with the data format desired. For more information please read the DOS READ reference.

If you don't want PRAGMA to append anything at the end of every write, you must specify NONE as the SEND TERM CHARACTER in PRAGMA.DES.

After having finished writing, the operating system file must always be closed by running DOS CLOSE WRITE.

Every one of these verbs will write the results of the operation whether it was successful or not into the standard noun IPC STATUS (IS).

For a list of all the possible IPC STATUS messages consult the IPC STATUS reference.

EXAMPLES

VERB called TEST DOS WRITE

```
1  COPY the value "Hello World" to the noun TEXT
2  DOS OPEN CREATE filename "TEST.TXT"
3  IF the value IPC STATUS "<>" the value "OK"
       do
4         GO TO line labeled ERROR
       else
       end
5  DOS WRITE the value TEXT
6  IF the value IPC STATUS "<>" the value "OK"
       do
7         GO TO line labeled ERROR
       else
       end
8  DOS CLOSE WRITE
9  IF the value IPC STATUS "<>" the value "OK"
       do
10        GO TO line labeled ERROR
       else
       end
11  RETURN
12  LABEL this line with
ERROR
13  DISPLAY the value IPC STATUS
```

It is important to always check the IPC STATUS since you are performing an operation on the operating system level where things are beyond the control of PRAGMA.

The verb creates a textfile TEST.TXT in the same directory where PRAGMA is, since no path is indicated. If the file TEST.TXT already exists, it will be overwritten and all it's contents will be lost. To avoid overwriting an existing file you can check if the file already exists with the verb DOS EXIST (DE).

The contents of the noun TEXT ("Hello World") will be written to the file TEST.TXT. At the end of the write the character specified in SEND TERM CHARACTER will be appended to the file, unless the character specified in SEND TERM CHARACTER is NONE, in which case nothing will be appended to the file following the the contents of the noun TEXT.

To append to an existing file you substitute in line two the verb DOS OPEN CREATE with the verb DOS OPEN APPEND. Every time you will then run the verb you will be appending the contents of the noun TEXT to the file TEST.TXT. Everything else will remain the same.

You can use the verb TEST DOS READ, shown in the reference of the verb DOS READ (DR), to see what you wrote with the verb TEST DOS WRITE.

~~~~~~~~~~~~~~~~~~~~~~~~

GET IPC STATUS GIS Standard Verb Operating System Operation
Every operation with a standard verb that accesses the operating system will always automatically perform a check if the operation was successful and write the result into the noun IPC STATUS.
The verb GET IPC STATUS puts the result of the latest operating system operation into the noun IPC STATUS. It has been included for compatibility with old applications.
GET IPC STATUS

SEE ALSO: IPC STATUS/ /
~~~~~~~~~~~~~~~~~~~~~~~~~~

IPC SERIAL ISE Standard Verb Input/Output Operation
Does absolutely nothing. In PRAGMA 3 this verb was necessary to redirect the flow of data from DOS to the serial port. PRAGMA 4 handles the situation differently and does not need to be redirected.
It has been included for compatibility with old applications.

It will always write "OK" into the noun IPC STATUS
IPC SERIAL

SEE ALSO: IPC STATUS/ /
~~~~~~~~~~~~~~~~~~~~~~~~~~

ADD + Standard Verb Arithmetic Operation
Adds two items and places the result in the target noun SUM.

If SUM contains the result of an earlier addition, it will be overwritten.

ADD the value ITEM to the value ITEM
Item can be a number or a noun that contains a number. Both items can have the same value. The addition does not alter the value(s) of the item.
SUM always contains the result of the most recent addition.
SUM itself can be used as an operand in an addition.

SEE ALSO: SUM/ /
~~~~~~~~~~~~~~~~~~~~~~~~~~

ATTRIBUTE BLINK ABL Standard Noun Noun

Used only in conjunction with the standard verb SET DISPLAY ATTRIBUTE (SDA).
Everything written on the screen after having issued this command will blink.
You can ADD any two attributes together to get the combined effect of both
attributes.
Works with both color and monochrome monitors.
To go back to normal you SET DISPLAY ATTRIBUTE with ATTRIBUTE NORMAL.
SET DISPLAY ATTRIBUTE to the attribute ATTRIBUTE BLINK

or

ADD the value ATTRIBUTE BLINK to the value ATTRIBUTE INVERSE
SET DISPLAY ATTRIBUTE to the attribute SUM (Monochrome only)
SEE ALSO: SET DISPLAY ATTRIBUTE/ATTRIBUTE NORMAL/ADD
~~~~~~~~~~~~~~~~~~~~~~~~~~

ATTRIBUTE BOLD   ABO Standard Noun Noun
Used only in conjunction with the standard verb SET DISPLAY ATTRIBUTE (SDA).
Everything written on the screen after having issued this command will be
bold.
Works with both color and monochrome monitors.
To go back to normal you SET DISPLAY ATTRIBUTE with ATTRIBUTE NORMAL.

SET DISPLAY ATTRIBUTE to the attribute ATTRIBUTE BOLD

SEE ALSO: SET DISPLAY ATTRIBUTE/ATTRIBUTE NORMAL/
~~~~~~~~~~~~~~~~~~~~~~~~~~

ATTRIBUTE INVERSE   AI   Standard Noun Noun
Used only in conjunction with the standard verb SET DISPLAY ATTRIBUTE (SDA).
Everything written on the screen after having issued this command will be
inverse.
Works only with monochrome monitors.
To go back to normal you SET DISPLAY ATTRIBUTE to ATTRIBUTE NORMAL.

SET DISPLAY ATTRIBUTE to the attribute ATTRIBUTE INVERSE

SEE ALSO: SET DISPLAY ATTRIBUTE/ATTRIBUTE NORMAL/
~~~~~~~~~~~~~~~~~~~~~~~~~~

ATTRIBUTE NORMAL   AN   Standard Noun Noun
Used only in conjunction with the standard verb SET DISPLAY ATTRIBUTE (SDA).
Everything written on the screen after having issued this command will be
normal.
It is used to reset the writing on the screen after having written with a
~~~~~~~~~~~~~~~~~~~~~~~~~~

different attribute, like for instance with ATTRIBUTE BOLD.
Works with both color and monochrome monitors.
SET DISPLAY ATTRIBUTE to the attribute ATTRIBUTE NORMAL

SEE ALSO: SET DISPLAY ATTRIBUTE/ATTRIBUTE BOLD/
~~~~~~~~~~~~~~~~~~~~~~~~~~

ATTRIBUTE UNDERLINE AU Standard Noun Noun
Used only in conjunction with the standard verb SET DISPLAY ATTRIBUTE (SDA).
Everything written on the screen after having issued this command will be
underlined.
Works only with monochrome monitors.
To go back to normal you SET DISPLAY ATTRIBUTE to ATTRIBUTE NORMAL.

SET DISPLAY ATTRIBUTE to the attribute ATTRIBUTE UNDERLINE

SEE ALSO: SET DISPLAY ATTRIBUTE/ATTRIBUTE NORMAL/
~~~~~~~~~~~~~~~~~~~~~~~~~~

BEGIN BG Standard Verb File Access Operation
Positions a file pointer directly ahead of the record referenced by NAME.
BEGIN merely positions a file pointer, so that the subsequent GET NEXT will
access the record. If the file does not contain a record with the referenced
expression, the GET NEXT statement will access the next existing record in the
file.

BEGIN at position NAME in file FILE

NAME identifies the record that you want to access. It can be a noun or an
expression.
FILE is the name that contains the record.

SEE ALSO: GET NEXT/BEGIN THIS/
DISCUSSION

BEGIN merely positions a file pointer; it does not GET or SAVE a record. The pointer is placed
directly ahead of the indicated record, so the subsequent GET NEXT will access the record.
Consider the following statements:

BEGIN at position NAME in file ADDRESSES
GET NEXT in file ADDRESSES

The first statement tells PRAGMA to position the file pointer ahead of the record referenced by NAME. The second statement tells PRAGMA to get the next record, which is the one referenced by NAME, provided that such a record exists.

The file you name does not actually have to contain a record with the referenced number or expression. The next GET NEXT will still access the next existing record in the file. For example, suppose the references in file ACCOUNTS are 600, 700, and 800. The following statements will result in record 700 being accessed.

```
BEGIN at position 650 in file ACCOUNTS
GET NEXT in file ACCOUNTS
```

It does not matter to PRAGMA that record 650 does not exist. The next record it gets will be record 700.

EXAMPLES

1. The following example shows the verb RUN ALL STATEMENTS. This verb prints a statement for each record in the file ACCOUNTS and prints out the totals for all statements. The verb BEGIN is used to position the file pointer ahead of the first record in the file ACCOUNTS. Records are accessed via GET NEXT in the verb STATEMENT LOOP; accessing continues until the end of the file is reached.

```
VERB  called  RUN ALL STATEMENTS
1     STATEMENT HEADING
2     CLEAR STATEMENT TOTALS
3     BEGIN FIRST in file ACCOUNTS
4     STATEMENT LOOP
5     SHOW STATEMENT TOTALS
```

Line 1 calls the verb STATEMENT HEADING; the verb prints a heading. Line 2 calls the verb CLEAR STATEMENT TOTALS; the verb clears all statement totals to zero, so new totals can be calculated. Line 3 contains the BEGIN FIRST statement that positions the file pointer ahead of the first record. Line 4 calls the verb STATEMENT LOOP, shown below.

The verb STATEMENT LOOP accesses each record in the file and prints it; the verb returns to RUN ALL STATEMENTS after printing the last record.

```
VERB  called  STATEMENT LOOP
1     GET NEXT in file  ACCOUNTS
2     STATEMENT PRINTING
3     REPEAT
```

2. A BEGIN statement can be used to position the file pointer at any record in a file. The following verb, which prints account statements, shows this.

```
VERB  called  RUN SOME STATEMENTS
1     DISPLAY LINE FEED
2     DISPLAY the value  "Start with account number: "
3     INPUT into the noun START#
4     DISPLAY the value "End with account number: "
5     INPUT into the noun END#
6     IF the value START# ">" the value END#
        do
          REPEAT
        else
        end
7     COMMENT text is If starting# = ending#, will do one statement.
8     BEGIN at position START# in file  ACCTS
9     STATEMENT HEADING
10    STATEMENT LOOP
11    SHOW STATEMENT TOTALS
```

Line 1 causes a carriage return and line feed. Lines 2-5 ask you for starting and ending account numbers. Line 6 tests whether the starting number is in the file; if it is not, lines 1-5 are repeated. Line 8 positions the file pointer at the record with the starting account number. Line 9 prints a statement heading. Line 10 calls the verb STATEMENT LOOP (shown next). Line 11 prints statement totals.

```
VERB  called  STATEMENT LOOP
1     GET NEXT in file ACCOUNT
2     IF the value ACC NO ">" the value END#
        do
          RETURN
        else
        end
3     STATEMENT PRINTING
4     REPEAT
```

Line 1 retrieves records, beginning with the starting account number and continuing through the ending account number. When the ending number is reached, line 2 returns to RUN SOME STATEMENTS. Line 3 prints statements. Line 4 repeats lines 1-3.

~~~~~~~~~~~~~~~~~~~~~~~~

```
BEGIN FIRST         BGF  Standard Verb  File Access Operation
BEGIN FIRST merely positions a file pointer ahead of the first record in a
file.
```
~~~~~~~~~~~~~~~~~~~~~~~~

BEGIN FIRST in file FILE

FILE is the name of the file that contains the record.

SEE ALSO: BEGIN/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

BEGIN LAST   BGL Standard Verb File Access Operation
BEGIN LAST merely positions a file pointer after the last record in a file.
A subsequent GET NEXT will hit the end of the file. A GET THIS will get the
last record of the file.

BEGIN LAST in file FILENAME

FILE is the name of the file that contains the record.

SEE ALSO: BEGIN/GET THIS/GET NEXT
~~~~~~~~~~~~~~~~~~~~~~~~~~

BEGIN NEXT   BGN Standard Verb File Access Operation
BEGIN NEXT merely positions a file pointer ahead of the next record in a file.
It is essentially a useless operation because the THIS pointer is always
before the NEXT record anyway.
This verb has been included merely for completeness.

BEGIN NEXT in file FILENAME

FILE is the name of the file that contains the record.

SEE ALSO: BEGIN/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

BEGIN THIS   BGN Standard Verb File Access Operation
BEGIN THIS positions a file pointer ahead of the present (THIS) record in a
file.
To get a preceding record in a file you do first a BEGIN THIS, followed by
a GET THIS.

BEGIN THIS in file FILE
GET THIS in file FILE

FILE is the name of the file that contains the record.
~~~~~~~~~~~~~~~~~~~~~~~~~

SEE ALSO: BEGIN/GET THIS/
~~~~~~~~~~~~~~~~~~~~~~~~

CLEAR SCREEN CS Standard Verb Screen Operation
Clears the screen, and in addition, also homes the cursor to row 1, column1, clears all character attributes, and resets the foreground/background colors to white on black.
This is useful as a way to clear any state the screen is to a known condition.

CLEAR SCREEN

SEE ALSO: ERASE SCREEN / /
~~~~~~~~~~~~~~~~~~~~~~~~~~

COLOR BLACK CBA Standard Noun Noun
Used in conjunction with the standard verbs SET FOREGROUND COLOR (SFC) and SET BACKGROUND COLOR (SBC). Everything written on the screen after having issued this command will have either a black foreground or background, provided the monitor accepts colors.
To change color you run the SET verbs again or you can run the verb CLEAR SCREEN (CS) to clear all screen attributes.
SET FOREGROUND COLOR to the color COLOR BLACK
SET BACKGROUND COLOR to the color COLOR BLACK

SEE ALSO: SET FOREGROUND COLOR/SET BACKGROUND COLOR/CLEAR SCREEN
~~~~~~~~~~~~~~~~~~~~~~~~

COLOR BLUE CBU Standard Noun Noun
Used in conjunction with the standard verbs SET FOREGROUND COLOR (SFC) and SET BACKGROUND COLOR (SBC). Everything written on the screen after having issued this command will have either a blue foreground or background, provided the monitor accepts colors.
To change color you run the SET verbs again or you can run the verb CLEAR SCREEN (CS) to clear all screen attributes.
SET FOREGROUND COLOR to the color COLOR BLUE
SET BACKGROUND COLOR to the color COLOR BLUE

SEE ALSO: SET FOREGROUND COLOR/SET BACKGROUND COLOR/CLEAR SCREEN
~~~~~~~~~~~~~~~~~~~~~~~~~~

COLOR CYAN CCY Standard Noun Noun
Used in conjunction with the standard verbs SET FOREGROUND COLOR (SFC) and SET BACKGROUND COLOR (SBC). Everything written on the screen after having issued this command will have either a cyan foreground or background, provided the monitor accepts colors.
To change color you run the SET verbs again or you can run the verb CLEAR SCREEN (CS) to clear all screen attributes.
SET FOREGROUND COLOR to the color COLOR CYAN
SET BACKGROUND COLOR to the color COLOR CYAN

SEE ALSO: SET FOREGROUND COLOR/SET BACKGROUND COLOR/CLEAR SCREEN
~~~~~~~~~~~~~~~~~~~~~~~~~

COLOR GREEN CG Standard Noun Noun
Used in conjunction with the standard verbs SET FOREGROUND COLOR (SFC) and SET BACKGROUND COLOR (SBC). Everything written on the screen after having issued this command will have either a green foreground or background, provided the monitor accepts colors.
To change color you run the SET verbs again or you can run the verb CLEAR SCREEN (CS) to clear all screen attributes.
SET FOREGROUND COLOR to the color COLOR GREEN
SET BACKGROUND COLOR to the color COLOR GREEN

SEE ALSO: SET FOREGROUND COLOR/SET BACKGROUND COLOR/CLEAR SCREEN
~~~~~~~~~~~~~~~~~~~~~~~~~

COLOR MAGENTA CMA Standard Noun Noun
Used in conjunction with the standard verbs SET FOREGROUND COLOR (SFC) and SET BACKGROUND COLOR (SBC). Everything written on the screen after having issued this command will have either a magenta foreground or background, provided the monitor accepts colors.
To change color you run the SET verbs again or you can run the verb CLEAR SCREEN (CS) to clear all screen attributes.
SET FOREGROUND COLOR to the color COLOR MAGENTA
SET BACKGROUND COLOR to the color COLOR MAGENTA

SEE ALSO: SET FOREGROUND COLOR/SET BACKGROUND COLOR/CLEAR SCREEN
~~~~~~~~~~~~~~~~~~~~~~~~~

COLOR RED CRD Standard Noun Noun
Used in conjunction with the standard verbs SET FOREGROUND COLOR (SFC) and SET BACKGROUND COLOR (SBC). Everything written on the screen after having
~~~~~~~~~~~~~~~~~~~~~~~~~

issued this command will have either a red foreground or background, provided the monitor accepts colors.
To change color you run the SET verbs again or you can run the verb CLEAR SCREEN (CS) to clear all screen attributes.
SET FOREGROUND COLOR to the color COLOR RED
SET BACKGROUND COLOR to the color COLOR RED

SEE ALSO: SET FOREGROUND COLOR/SET BACKGROUND COLOR/CLEAR SCREEN
~~~~~~~~~~~~~~~~~~~~~~~~~~

COLOR WHITE CW Standard Noun Noun
Used in conjunction with the standard verbs SET FOREGROUND COLOR (SFC) and SET BACKGROUND COLOR (SBC). Everything written on the screen after having issued this command will have either a white foreground or background, provided the monitor accepts colors.
To change color you run the SET verbs again or you can run the verb CLEAR SCREEN (CS) to clear all screen attributes.
SET FOREGROUND COLOR to the color COLOR WHITE
SET BACKGROUND COLOR to the color COLOR WHITE

SEE ALSO: SET FOREGROUND COLOR/SET BACKGROUND COLOR/CLEAR SCREEN
~~~~~~~~~~~~~~~~~~~~~~~~~~

COLOR YELLOW CY Standard Noun Noun
Used in conjunction with the standard verbs SET FOREGROUND COLOR (SFC) and SET BACKGROUND COLOR (SBC). Everything written on the screen after having issued this command will have either a yellow foreground or background, provided the monitor accepts colors.
To change color you run the SET verbs again or you can run the verb CLEAR SCREEN (CS) to clear all screen attributes.
SET FOREGROUND COLOR to the color COLOR YELLOW
SET BACKGROUND COLOR to the color COLOR YELLOW

SEE ALSO: SET FOREGROUND COLOR/SET BACKGROUND COLOR/CLEAR SCREEN
~~~~~~~~~~~~~~~~~~~~~~~~~~

COMMENT CM Standard Verb Definition Operation
Used for documentation within a verb of file definition. A COMMENT instruction is not executed by PRAGMA.
A COMMENT can be of unlimited length and the format of your comment is up to you. By typing spaces at the start of each line, for instance, you can create an indented or blocked comment. You may also use the underline character if you wish to explain something about a particular control character.
~~~~~~~~~~~~~~~~~~~~~~~~~~

COMMENT text is YOUR TEXT

YOUR TEXT can be any text you like.

SEE ALSO: / /
~~~~~~~~~~~~~~~~~~~~~~~~

COPY   CP   Standard Verb Miscellaneous Data Manipulation
Copies the contents of a noun, number or expression into a specified noun.
The verb COPY creates a copy of the original noun without affecting its con-
tents in any way.
You cannot COPY a verb or a file.

COPY the value ITEM to the noun NOUN

ITEM can be a noun, number or expression. NOUN must be a noun.

SEE ALSO: VERB COPY / /
~~~~~~~~~~~~~~~~~~~~~~~~~~

CURSOR COLUMN   CC   Standard Noun Noun
In order to use the verb SET CURSOR POSITION (SCP), first COPY to the nouns
CURSOR COLUMN and CURSOR ROW the desired location on the screen where you want
the cursor to be. The home position (upper left corner) is row 1, column 1.

The verb GET CURSOR POSITION (GCP) will write to the nouns CURSOR COLUMN and
CURSOR ROW the current location of the cursor on the screen.
COPY the value ITEM to the noun CURSOR COLUMN
ITEM must be a number or a noun containing a number.

DISPLAY the value CURSOR COLUMN

SEE ALSO: SET CURSOR POSITION/GET CURSOR POSITION/CURSOR ROW
~~~~~~~~~~~~~~~~~~~~~~~~

CURSOR ROW   CR   Standard Noun Noun
In order to use the verb SET CURSOR POSITION (SCP), first COPY to the nouns
CURSOR COLUMN and CURSOR ROW the desired location on the screen where you want
the cursor to be. The home position (upper left corner) is row 1, column 1.

The verb GET CURSOR POSITION (GCP) will write to the nouns CURSOR COLUMN and
CURSOR ROW the current location of the cursor on the screen.
COPY the value ITEM to the noun CURSOR ROW
ITEM must be a number or a noun containing a number.

DISPLAY the value CURSOR ROW
~~~~~~~~~~~~~~~~~~~~~~~~

SEE ALSO: SET CURSOR POSITION/GET CURSOR POSITION/CURSOR COLUMN
~~~~~~~~~~~~~~~~~~~~~~~~~

CUT CT Standard Verb Miscellaneous Data Manipulation
Cuts a number, an expression or the contents of a noun at a point indicated by another number, expression or noun. Puts the results into the target nouns HEAD, TAIL and LENGTH. CUT has no effect on the original item. If PRAGMA cannot find the cut point, no cut is made and the contents of the original is written into TAIL. HEAD will contain nothing or NULL (""). LENGTH will be 0.
You can CUT after a number of characters or after a searched for string.
CUT the value ITEM after CUT POINT

ITEM and CUT POINT can be a number, noun or expression. The cut is made after the place indicated by the number or character(s) indicated by CUT POINT.
HEAD contains the left-hand result, TAIL the right-hand result of the cut.
LENGTH contains a number of how many characters are in HEAD after the cut.
SEE ALSO: SPLIT/JOIN/
DISCUSSION

CUT has no effect on the original item. It copies from the original item into HEAD and TAIL and leaves the original intact.

Since HEAD, TAIL and LENGTH are target nouns, if you cut them their original contents are lost. For example, if the content of HEAD after a CUT operation is the expression ABCD and you instruct PRAGMA to CUT HEAD after 2, the content of HEAD will be AB, and TAIL will contain CD. LENGTH will be 2.

Also, bear in mind that LENGTH is a target noun for INPUT, INPUT 0, $INPUT, RECEIVE, and JOIN. Any of these verbs could overwrite the contents of LENGTH.

If PRAGMA cannot find the cut point, no cut is made and the entire contents of the original are written into TAIL. HEAD will contain nothing, or Null (""). LENGTH will be zero. This occurs if the cut point is an expression that does not exist in the original item.

If the cut point is indicated by a number that contains a decimal (such as 5.5), the decimal will be ignored. For example, the instruction

CUT the value RENEWAL after 5.5

gives the following results:

HEAD will contain the first five characters of RENEWAL
TAIL will contain the remaining characters of RENEWAL
LENGTH will contain the integer 5

EXAMPLES

1. CUTTING AFTER AN EXPRESSION
~~~~~~~~~~~~~~~~~~~~~~~~~

In this example PRAGMA is instructed to cut the noun EMPLOYEE NAME after the comma. Since the comma is in quotes, it is treated as an expression.

CUT the value EMPLOYEE NAME after ","

The following table shows the results of this instruction, depending on the content of the original item. Brackets ( ] and [ ) are included in the HEAD and TAIL columns to show the exact contents of HEAD and TAIL, including spaces.

EMPLOYEE NAME	HEAD	TAIL	LENGTH
Smith, John	]Smith,[	] John[	6
Smith,John	]Smith,[	]John[	6
Smith,John,Jr.	]Smith,[	]John,Jr.[	6
Only one cut after 1st comma.			
Smith John	""(null)	]Smith John[	0
No comma in the expression.			
Smith John,	]Smith John,[	""(null)	15
No characters after cut point,so TAIL is empty.			

2. CUTTING AFTER A SPECIFIED NUMBER OF CHARACTERS

In the following example, an eight-digit number is cut twice to extract the fourth and fifth digits. The noun ACCOUNT NO contains the eight digits. The fourth and fifth digits represent the customer's credit limit, in hundreds of dollars.

```
VERB  called  CREDIT LIMIT CHECK

1     CUT the value ACCOUNT NO  after  3
2     CUT the value TAIL  after  2
3     MULTIPLY the value HEAD by the value 100
4     COPY the value PRODUCT to the noun CREDIT LIMIT
5     LOOKUP CURRENT BALANCE
6     ADD the value SALE AMOUNT to the value CURRENT BALANCE
7     IF the value SUM ">" the value CREDIT LIMIT
        do
          OVER CREDIT LIMIT
        else
        end
```

Following line 1, the first three digits of ACCOUNT NO are in HEAD, the last five digits are in TAIL, and LENGTH contains the number 3. After the cut in line 2, the fourth and fifth digits of ACCOUNT NO, which indicate the credit limit, are in HEAD, and the sixth an seventh digits of ACCOUNT NO are in TAIL. LENGTH contains the number 2. In line 3, the number in HEAD is multiplied by 100 to convert the limit to hundreds of dollars. Line 5 obtains the customer's current balance. Lines 6 and 7 add the value of the current sale to the balance and check the total against the credit limit; if the total exceeds the credit limit, the routine OVER CREDIT LIMIT is called.

3.1 DUMMY CUTS (NULL CUTS)

There may be times when you want to display or print a number in a left-aligned, no-comma format (as if it were an expression). For instance, you might want to display the number] 123,456,789[in the form]123456789 [(brackets indicate column boundaries).

You can do this with a dummy CUT operation, as shown in the next two examples. (The noun POPULATION contains the number 123,456,789.)

```
CUT the value 123,456,789 after ""

CUT the value POPULATION after ""
```

Both instructions result in 123456789 being written to HEAD, and "" (null) to TAIL. LENGTH will contain 9. The number can now be displayed in left-aligned format using the layout specification. Remember that the results of a CUT (in HEAD and TAIL) are always expressions:

```
DISPLAY LAYOUT using layout 13 the value HEAD
```

The number will be displayed in this way:

```
]123456789    [
```

3.2

A dummy cut can also be used to determine how many characters are contained in a noun. This could be used to center a heading of unknown length, as shown in the following example. This example also illustrates the CUT operation after a noun that contains a number.

```
VERB called CENTER HEADING

1     CUT the value HEADING after ""
2     SUBTRACT the value of LENGTH from the value of 80
3     DIVIDE the value DIFFERENCE by 2
4     SPLIT the value QUOTIENT
```

```
5     CUT the value 80 SPACES after INTEGER
6     PRINT the value HEAD
7     PRINT the value HEADING
8     PRINTER LINE FEED
```

After PRAGMA executes the dummy CUT in line 1, the entire contents of HEADING go into HEAD, TAIL is empty (""), and the noun LENGTH contains a number showing the number of characters in HEAD. To determine how many spaces to print before printing the noun HEADING, LENGTH is subtracted from the page width (that is, from 80), and the result is divided by two. This is done in lines 2 and 3. PRINTER LINE FEED, in line 8, causes the printer to output a line feed.

~~~~~~~~~~~~~~~~~~~~~~~~

DELETE DL Standard Verb File Access Operation
Removes a record with the specified reference from a file. If PRAGMA cannot find the specified record, the EXTERNAL ECHO will contain a "record not found" message and no record will be deleted.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.
DELETE reference CHOICE in file FILE

CHOICE can be a noun, number or an expression. CHOICE is the reference of the record. FILE is the name of the file that contains the record to be deleted.

SEE ALSO: DELETE ALL/DELETE THIS/
~~~~~~~~~~~~~~~~~~~~~~~~

DELETE ALL DA Standard Verb File Access Operation
Deletes all the records in a file.
It does not affect the definition of the file.

The noun EXTERNAL ECHO contains the status message of the DELETE ALL operation.
Operates very fast. It is the best way to clear a file of records.
DELETE ALL in file FILE

FILE is the name of the file whose records are to be deleted.

SEE ALSO: DELETE/DELETE THIS/EXTERNAL DELETE ALL
DELETE ALL and EXTERNAL DELETE ALL delete all the records in a file, without affecting the definition of the file.

These two verbs operate very fast and this is the best way to clear a file of all its records.

The noun EXTERNAL ECHO or FILE STATUS contains the status message of the DELETE ALL or EXTERNAL

DELETE ALL operations.

If any record has been locked by another user you will get back the status "conflict" in the NOUN EXTERNAL ECHO or a 3 in the noun FILE STATUS and the file will not be deleted.

If you are using the PRAGMA 3 internal emulation mode (mode 1) DELETE ALL and EXTERNAL DELETE ALL will wait until nobody has any pending locks and then proceed.

WARNING

DELETE_ALL and EXTERNAL DELETE ALL function by performing a create operation. This causes a problem if under DOS you use the DOS command APPEND and put your PFM or Btrieve files into a different directory than the generated .EXE file. When PFM or Btrieve look for a file, APPEND searches for the file and all is well. But when a DELETE_ALL performs a create operation, it automatically does it in the same directory where the program resides. And from then on all data is written and read from that newly created file.

BTRIEVE WARNING

When Btrieve attempts a DELETE ALL or an EXTERNAL DELETE ALL and encounters a locked record it may give an "INTERNAL LOGIC ERROR" instead of a "conflict". More testing is needed to clarify this situation.

~~~~~~~~~~~~~~~~~~~~~~~~~

DELETE FIRST DLF Standard Verb File Access Operation
Deletes the first record in a file.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

DELETE FIRST in file FILE

FILE is the name of the file whose FIRST record is to be deleted.

SEE ALSO: DELETE/DELETE THIS/
~~~~~~~~~~~~~~~~~~~~~~~~~

DELETE LAST DLL Standard Verb File Access Operation
Deletes the last record in a file.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

DELETE LAST in file FILE

FILE is the name of the file whose LAST record is to be deleted.

SEE ALSO: DELETE/DELETE THIS/
~~~~~~~~~~~~~~~~~~~~~~~~~~

DELETE NEXT DLN Standard Verb File Access Operation
Deletes the next record in a file.
This is particularly useful after having positioned the file pointer directly
ahead of a record with the verb BEGIN.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.
DELETE NEXT in file FILE

FILE is the name of the file whose NEXT record is to be deleted.

SEE ALSO: BEGIN/DELETE/
~~~~~~~~~~~~~~~~~~~~~~~~

DELETE THIS DLT Standard Verb File Access Operation
Deletes the present record in a file.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

DELETE THIS in file FILE

FILE is the name of the file whose THIS record is to be deleted.

SEE ALSO: DELETE/DELETE ALL/
~~~~~~~~~~~~~~~~~~~~~~~~~~

DIFFERENCE DF Standard TargetNoun
Used with the standard verb SUBTRACT.
Contains the result of the most recent subtraction operation or the most
recent value copied to it.
~~~~~~~~~~~~~~~~~~~~~~~~~~

SEE ALSO: SUBTRACT/ /
~~~~~~~~~~~~~~~~~~~~~~~~

DISPLAY DS Standard Verb Input/Output Operation
Shows text and numbers on your screen.
The output is displayed from left to right, starting in the first available column and continuing through column 80. The overflow is wrapped to the next line. Characters from successive DISPLAY instructions are displayed directly one after another on the same line until the line is full.
Control characters and cursor positioning verbs format the DISPLAY output.
DISPLAY the value ITEM

Item is the noun or expression you want to display. Expressions must be preceded by a double quote ("). Control characters must be entered as expressions.

SEE ALSO: DISPLAY LAYOUT/SET CURSOR POSITION/SET DISPLAY ATTRIBUTE
~~~~~~~~~~~~~~~~~~~~~~~~~~

DISPLAY LAYOUT DSL Standard Verb Input/Output Operation
Shows text and numbers on your screen formatted as a column of a particular width. Expressions are displayed left-aligned in the column, numbers right-aligned in the column, with or without decimal point. Any unused spaces in the column are filled with the space (blank) character. When 80 characters have been displayed on a line, the 81st character is displayed at the beginning of the next line.
DISPLAY using layout LAYOUT the value ITEM

LAYOUT is the layout specification indicating the format in which the noun, number or expression named in ITEM is to be displayed. A LAYOUT of 20 will establish a column of the width of 20 characters. To layout several columns on a line, use several DISPLAY LAYOUT instructions.
SEE ALSO: DISPLAY/SET CURSOR POSITION/SET DISPLAY ATTRIBUTE
You may also enter a layout like 0.2. This will cause the value to be output in a field width that is the exact size of the number, with commas added, etc.

~~~~~~~~~~~~~~~~~~~~~~~~

DIVIDE / Standard Verb Arithmetic Operation
Performs a division with two numbers, a number and a noun containing a number or two nouns containing numbers.

The result of the division is placed into the target noun QUOTIENT.

DIVIDE the value ITEM by ITEM

ITEM can be a number or a noun containing a number.
~~~~~~~~~~~~~~~~~~~~~~~~

SEE ALSO: QUOTIENT/ /
~~~~~~~~~~~~~~~~~~~~~~~~

ERASE SCREEN   ES   Standard Verb Screen Operation
Clears the screen and homes the cursor to row 1, column 1, but leaves the attributes and colors as they are.
This is useful when displaying the next page in a multiple page document or anytime when you only want to erase the text on the screen, but leave the character qualities as they are.
Also used to paint the screen background with a color.
ERASE SCREEN

SET BACKGROUND COLOR to the color COLOR BLUE
ERASE SCREEN
Will paint the whole screen with a blue background color.

SEE ALSO: CLEAR SCREEN/ /
~~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL BEGIN   XB   Standard Verb External File Access Operation
Positions a file pointer directly ahead of the record referenced by NAME.
EXTERNAL BEGIN merely positions a file pointer, so that the subsequent EXTERNAL GET NEXT will access the record. If the file does not contain a record with the referenced expression, the EXTERNAL GET NEXT statement will access the next existing record in the file.

EXTERNAL BEGIN at position NAME in file FILE

NAME identifies the record that you want to access. It can be a noun or an expression.
FILE is the name that contains the record.

SEE ALSO: EXTERNAL GET NEXT/EXTERNAL BEGIN THIS/
~~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL BEGIN FIRSTXBF Standard Verb External File Access Operation
EXTERNAL BEGIN FIRST merely positions a file pointer ahead of the first record in a file.

EXTERNAL BEGIN FIRST in file FILE

FILE is the name that contains the record.

SEE ALSO: EXTERNAL GET/EXTERNAL BEGIN/
~~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL BEGIN LAST XBL Standard Verb External File Access Operation
EXTERNAL BEGIN LAST merely positions a file pointer after the last record in
a file. A subsequent EXTERNAL GET NEXT will hit the end of the file.
An EXTERNAL GET THIS will get the last record of the file.

EXTERNAL BEGIN LAST in file FILE

FILE is the name that contains the record.

SEE ALSO: EXTERNAL BEGIN/EXTERNAL GET THIS/EXTERNAL GET NEXT
~~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL BEGIN NEXT XBN Standard Verb External File Access Operation
EXTERNAL BEGIN NEXT merely positions a file pointer ahead of the next
record in a file.
It is essentially a useless operation because the THIS pointer is always
before the NEXT record anyway.
This verb has been included merely for completeness.

EXTERNAL BEGIN NEXT in file FILE

FILE is the name that contains the record.

SEE ALSO: EXTERNAL GET/EXTERNAL BEGIN/
~~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL BEGIN THIS XBN Standard Verb External File Access Operation
EXTERNAL BEGIN THIS positions a file pointer ahead of the present (THIS)
record in a file.
To get a preceding record in a file you do first a EXTERNAL BEGIN THIS,
followed by an EXTERNAL GET THIS.

EXTERNAL BEGIN THIS in file FILE
EXTERNAL GET THIS in file FILE

FILE is the name of the file that contains the record.

SEE ALSO: EXTERNAL GET THIS/EXTERNAL BEGIN/
~~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL DELETE   XD   Standard Verb External File Access Operation
Removes a record with the specified reference from a file. If PRAGMA cannot
find the specified record, the EXTERNAL ECHO will contain a "record not found"
~~~~~~~~~~~~~~~~~~~~~~~~~~

message and no record will be deleted.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.
EXTERNAL DELETE reference CHOICE in file FILE

CHOICE can be a noun, number or an expression. CHOICE is the reference of the record. FILE is the name of the file that contains the record to be deleted.

SEE ALSO: EXTERNAL GET THIS/EXTERNAL BEGIN/
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL DELETE FIRSXDF Standard Verb External File Access Operation
Deletes the first record in a file.

The reference of the record is written to the target noun REFERENCE,
The noun EXTERNAL ECHO contains the status message of the file operation.

EXTERNAL DELETE FIRST in file FILE

FILE is the name of the file that contains the record to be deleted.

SEE ALSO: EXTERNAL DELETE/EXTERNAL DELETE THIS/
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL DELETE LASTXDL Standard Verb External File Access Operation
Deletes the last record in a file.

The reference of the record is written to the target noun REFERENCE,
The noun EXTERNAL ECHO contains the status message of the file operation.

EXTERNAL DELETE LAST in file FILE

FILE is the name of the file whose LAST record is to be deleted.

SEE ALSO: EXTERNAL DELETE/EXTERNAL DELETE THIS/
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL DELETE NEXTXDN Standard Verb External File Access Operation
Deletes the next record in a file. This is particularly useful after having positioned the file pointer directly ahead of a record with the verb EXTERNAL BEGIN.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL DELETE NEXT in file FILE

FILE is the name of the file whose NEXT record is to be deleted.

SEE ALSO: EXTERNAL BEGIN/EXTERNAL DELETE /

EXTERNAL DELETE THISXDT Standard Verb External File Access Operation
Deletes the present record in a file.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

EXTERNAL DELETE THIS in file FILE

FILE is the name of the file whose THIS record is to be deleted.

SEE ALSO: EXTERNAL DELETE / /

EXTERNAL ECHO   XE   Standard TargetNoun
Used with the verbs that perform file operations (external and regular).
It contains the status message of the most recent file operation. This status
message tells you whether or not the file operation completed properly.
Both EXTERNAL ECHO and FILE STATUS are set after file operations, and either
may be checked. EXTERNAL ECHO contains text which may be different for each
country, being dependent on the file status messages selected in PRAGMA.DES.
DISPLAY the value EXTERNAL ECHO

SEE ALSO: FILE STATUS/IPC STATUS/
The noun EXTERNAL ECHO (XE) will contain a message after all file operations (except those using mock internal emulation; mockflag = 1).

Both FILE STATUS and EXTERNAL ECHO are set after file operations, and either may be checked.

FILE STATUS, since it contains numeric values, is language independent; whereas EXTERNAL ECHO contains text which is different for each country.

If EXTERNAL ECHO is used, the verb is dependent upon the file status messages selected in PRAGMA.DES. That means that for instance, when you check with an IF statement if the EXTERNAL ECHO is equal to "record not found", your program will not work if in PRAGMA.DES instead of "record not found" you have "scheda non trovata".

The following are the possible messages of EXTERNAL ECHO in the standard American version of PRAGMA:

OK
record not found
duplicate reference
conflict
file is locked
queue is empty
queue is locked
prepare to stop
NO SUCH FILE
REFERENCE TOO LONG
MORE NOUNS IN LOCAL FILE
FEWER NOUNS IN LOCAL FILE
NO SUCH QUEUE
NO RESPONSE RECEIVED
EXTERNAL DISC IS FULL
TRANSMISSION ERROR
INTERNAL LOGIC ERROR
UNKNOWN EXTERNAL ECHO?

Be aware that the messages are case sensitive. The old fashioned way of spelling DISC is a leftover from the old days.

OK

The communication between PRAGMA and the filemanager was successful. The majority of your communications will result in this message. If EXTERNAL ECHO does not contain an "OK", then something special will usually need to be done to handle or correct the situation.

record not found

A file record with the requested reference was not found.

duplicate reference

The filemanager tried to save a record and found that a record with an identical reference already exists in the specified file. The record was not saved.

A workstation attempted to lock a record more than once.

conflict

A workstation tried to retrieve a file record that was controlled at the time by another workstation.

file is locked

The specified Fileserver file has been locked out of use by the Fileserver console operator.

queue is empty

The specified Fileserver message queue is empty.

queue is locked

The specified Fileserver message queue has been locked out of use by the Fileserver console operator.

prepare to stop

The Fileserver console operator has selected the PREPARE TO STOP function to advise the workstations that the Fileserver will be shutting down.

NO SUCH FILE

The filemanager tried to use a file that was defined in the VOCAB but not in the operating system or at the Fileserver.

REFERENCE TOO LONG

You tried to use a reference longer than 32,767 characters.
The record being accessed is bigger than the RECORD SIZE stated in PRAGMA.DES.
A reference longer than 255 bytes is being used with Btrieve.

MORE NOUNS IN LOCAL FILE

The number of nouns used in your workstation file definition is more than the number of values per record in the Fileserver file or more than the number of values per message in the Fileserver queue.

FEWER NOUNS IN LOCAL FILE

The number of nouns used in your workstation file definition is less than the number of values per record in the Fileserver file or less than the number of values per message in the Fileserver queue.

NO SUCH QUEUE

You tried to use a message queue that is defined as a file at your workstation, but is not defined

as a message queue at the Fileserver.

NO RESPONSE RECEIVED

If you are using Btrieve, Btrieve itself has not been loaded.
An incorrect selection of file manager has been made in PRAGMA.DES.
The PRAGMA Fileserver does not respond to your workstation after several seconds of requests.

EXTERNAL DISC IS FULL

You tried to store a record or message when there was no more free space on the hard disk.

TRANSMISSION ERROR

A workstation cannot receive a correct transmission from the Fileserver.

INTERNAL LOGIC ERROR

The filemanager attempted to access a file unsuccessfully.

If you are using Btrieve as filemanager and the result of a file operation is "INTERNAL LOGIC ERROR" for the EXTERNAL ECHO, the FILE STATUS will contain the Btrieve error number. This is to help to trace Btrieve errors.

UNKNOWN EXTERNAL ECHO?

The workstation logic and the Fileserver logic are not fully compatible.

Two more external echoes mentioned in some PRAGMA 3 manuals must also be mentioned.

RESET FAILS

It is an obsolete message that was only used by TINA.

unsafe save

When you attempt to do a SAVE THIS, PRAGMA 3 checks to see if the record you are saving is locked to you. The FILESERVER does not care. PRAGMA 3 goes ahead and performs the SAVE THIS in either case. If the FILESERVER answers with an "OK" and the record was not locked to you, PRAGMA 3 changes this to "unsafe save", to warn you of a potentially bad situation. PRAGMA 4 does not worry about this and you just get the "OK".

~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET XG Standard Verb External File Access Operation
Retrieves a record from a file. If PRAGMA cannot find the specified record,
the EXTERNAL ECHO will contain a "record not found" message and no record
in the file will be retrieved.

After accessing the file the reference of the record is written to the target
noun REFERENCE. EXTERNAL ECHO contains the status message of the access.
EXTERNAL GET reference CHOICE in file FILE

CHOICE can be a noun, number or an expression that identifies the record you
want to access. FILE is the name of the file that is to be accessed.

SEE ALSO: EXTERNAL BEGIN/EXTERNAL GET THIS/
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET FIRST XGF Standard Verb External File Access Operation
Retrieves the first record from a file.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

EXTERNAL GET FIRST in file FILE

FILE is the name of the file whose FIRST record is to be accessed.

SEE ALSO: EXTERNAL GET/EXTERNAL BEGIN/
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET LAST XGL Standard Verb External File Access Operation
Retrieves the last record from a file.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

EXTERNAL GET LAST in file FILE

FILE is the name of the file whose LAST record is to be accessed.

SEE ALSO: EXTERNAL GET/EXTERNAL BEGIN/EXTERNAL GET FIRST
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET NEXT XGN Standard Verb External File Access Operation
Retrieves the next record from a file.

This is particularly useful after having positioned the file pointer directly
ahead of a record with the verb EXTERNAL BEGIN.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.
EXTERNAL GET NEXT in file FILE

FILE is the name of the file whose NEXT record is to be accessed.

SEE ALSO: EXTERNAL GET/EXTERNAL BEGIN/
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET THIS XGT Standard Verb External File Access Operation
Retrieves the present record from a file.
This is particularly useful after having positioned the file pointer directly
ahead of a record with the verb EXTERNAL BEGIN THIS, enabling you to retrieve
the previous record in a file.
The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.
EXTERNAL GET THIS in file FILE

FILE is the name of the file to be accessed.

SEE ALSO: EXTERNAL BEGIN THIS/EXTERNAL GET/
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL RELEASE XR Standard Verb External File Access Operation
This verb tells the workstation to let go of the file record that it curren-
tly controls. This verb works only when PRAGMA's Fileserver is chosen as
filemanager for compatibility with the old record locking scheme.
Although another file operation releases a record, this verb is useful to
free a record when you only want to look at it and don't want to hold up
some other user that might want to access the same record.
EXTERNAL RELEASE

SEE ALSO: RELEASE/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL SAVE XS Standard Verb External File Access Operation
Writes a new record and reference to the specified file.
The REFERENCE is not important to be specified during this operation, since it
has been defined in the file definition.
After a successful EXTERNAL SAVE operation the reference for the saved record
is written to the target noun REFERENCE.
~~~~~~~~~~~~~~~~~~~~~~~~~

The noun EXTERNAL ECHO contains the status message of the file operation.
EXTERNAL SAVE reference REFERENCE in file FILE

REFERENCE can be anything since it is not used by this operation.
FILE is the name of the file where the record is written to.

SEE ALSO: EXTERNAL BEGIN THIS/EXTERNAL GET/
~~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL SAVE FIRST XSF Standard Verb External File Access Operation
Updates the first record in the specified file.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

EXTERNAL SAVE FIRST in file FILE

FILE is the name of the file where the record is written to.

SEE ALSO: EXTERNAL SAVE/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL SAVE LAST XSL Standard Verb External File Access Operation
Updates the last record in the specified file.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

EXTERNAL SAVE LAST in file FILE

FILE is the name of the file where the record is written to.

SEE ALSO: EXTERNAL SAVE/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL SAVE NEXT XSN Standard Verb External File Access Operation
Updates the next record in the specified file.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

EXTERNAL SAVE NEXT in file FILE

FILE is the name of the file where the record is written to.
~~~~~~~~~~~~~~~~~~~~~~~~~

SEE ALSO: EXTERNAL SAVE/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL SAVE THIS XST Standard Verb External File Access Operation
Updates the present record in the specified file.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

EXTERNAL SAVE THIS in file FILE

FILE is the name of the file where the record is written to.

SEE ALSO: EXTERNAL SAVE/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL TIME   XT   Standard Verb Pragma Fileserver Access Operation
This verb tells the workstation to read PRAGMA's Fileserver clock.
It is only used when PRAGMA's Fileserver is specified as external file manager.

The noun EXTERNAL ECHO contains the status message of the file operation.
EXTERNAL TIME into the noun NOUN

The fileserver will copy a string of 20 digits and spaces into the noun NOUN
in the following format: YYYY MM DD hh mm ss

SEE ALSO: PUT TIME/ /
~~~~~~~~~~~~~~~~~~~~~

FRACTION   FC   Standard TargetNoun
Used with the verb SPLIT

Contains the fractional (decimal) result of the most recent SPLIT operation
or the most recent value copied to it.

SPLIT the value of ITEM
DISPLAY the value FRACTION

ITEM can be a number or a noun containing a number.

SEE ALSO: SPLIT/INTEGER/
~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~

GET GT Standard Verb File Access Operation
Retrieves a record from a file. If PRAGMA cannot find the specified record, the EXTERNAL ECHO will contain a "record not found" message and no record in the file will be retrieved.

After accessing the file the reference of the record is written to the target noun REFERENCE. EXTERNAL ECHO contains the status message of the access.
GET reference CHOICE in file FILE

CHOICE can be a noun, number or an expression that identifies the record you want to access. FILE is the name of the file that is to be accessed.

SEE ALSO: BEGIN/GET THIS/
~~~~~~~~~~~~~~~~~~~~~~~~~

GET CURSOR POSITION GCP Standard Verb Screen Operation
This verb reports the current cursor position. It sets the nouns CURSOR ROW and CURSOR COLUMN to the current location of the cursor on the screen. Note that the contents of these nouns will only be correct immediately after each time this verb is called; they will not be continuously updated automatically as the cursor is moved.

GET CURSOR POSITION

SEE ALSO: CURSOR ROW/CURSOR COLUMN/SET CURSOR POSITION
~~~~~~~~~~~~~~~~~~~~~~~~~

GET FIRST GTF Standard Verb File Access Operation
Retrieves the first record from a file.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

GET FIRST in file FILE

FILE is the name of the file whose FIRST record is to be accessed.

SEE ALSO: GET/BEGIN/
~~~~~~~~~~~~~~~~~~~~~~~~~

GET LAST GTL Standard Verb File Access Operation
Retrieves the last record from a file.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

GET LAST in file FILE

FILE is the name of the file whose LAST record is to be accessed.

SEE ALSO: GET/BEGIN/GET FIRST
~~~~~~~~~~~~~~~~~~~~~~~~~~

GET NEXT GTN Standard Verb File Access Operation
Retrieves the next record from a file.
This is particularly useful after having positioned the file pointer directly ahead of a record with the verb BEGIN.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.
GET NEXT in file FILE

FILE is the name of the file whose NEXT record is to be accessed.

SEE ALSO: GET/BEGIN/
~~~~~~~~~~~~~~~~~~~~~~~~~~

GET THIS GTT Standard Verb File Access Operation
Retrieves the present record from a file.
This is particularly useful after having positioned the file pointer directly ahead of a record with the verb BEGIN THIS, enabling you to retrieve the previous record in a file.
The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.
GET THIS in file FILE

FILE is the name of the file to be accessed.

SEE ALSO: BEGIN THIS/GET/
~~~~~~~~~~~~~~~~~~~~~~~~~~

GO TO GO Standard Verb Execution Flow Control
Instructs PRAGMA to jump to another line in the same verb that is marked with a label with the same name of the GO TO instruction. Can be used to jump forward or backward, but not to another verb.
If during the definition of a verb a GO TO does not have a corresponding label, a warning message is displayed. You cannot exit the definition of a verb
~~~~~~~~~~~~~~~~~~~~~~~~~~

if a label is missing.
GO TO line labeled LABEL   and if the LABEL does not exist
Warning. This label does not currently exist.

LABEL is the label assigned to the line you want PRAGMA to jump to.

SEE ALSO: LABEL/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

HEAD   HD   Standard TargetNoun
Used with the verbs CUT and JOIN.

Contains the result of the most recent JOIN operation, the left-handed result
of the most recent CUT operation, or the most recent value copied to it.

SEE ALSO: CUT/JOIN/
~~~~~~~~~~~~~~~~~~~~~~~~~

IF   IF   Standard Verb Conditional Operation
Compares two items for equality, inequality, or relative value. If the condi-
tion tested is true, the DO portion of the verb is exectuted. If the conditi-
on tested is not true, the ELSE portion is executed. You can test for two
conditions simultaneosly or nest IF verbs an unlimited number of times.
The allowed comparison must be an expression or a noun containing one of the
following strings: "<", ">", "=", "^", "<=", ">=", "=>" ,"=<", "<>", "><"
IF the value ITEM 1 CONDITION ITEM 2
     do   VERB(S)
     else VERB(S)
     end

ITEM 1 and ITEM 2 are the nouns, numbers or expressions you want to compare.
SEE ALSO: IF REFERENCE/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

IF REFERENCE   IR   Standard Verb Conditional Operation
Compares an item with the references in the index of a specified file. If the
condition tested is true, the DO portion of the verb is executed. If the con-
dition tested is not true, the ELSE portion is executed. You can test for two
conditions simultaneously or nest IF verbs an unlimited number of times.
The allowed comparison must be an expresion or noun containing one of the
following strings: "<", ">", "=". "^", "<=", ">=", "=>", "=<", "<>", "><"
IF REFERENCE the value ITEM CONDITION the records in file FILE
     do   VERB(S)
     else   VERB(S)
~~~~~~~~~~~~~~~~~~~~~~~~~

```
     end

ITEM is the noun, number or expression to compare with the file references.
SEE ALSO: IF/ /
~~~~~~~~~~~~~~~~~~~~~~~~~
```

INPUT IN Standard Verb Input/Output Operation
Accepts input from the keyboard of a maximum of 512 characters and writes it into a noun you designate. INPUT accepts spaces, letters, digits, punctuation marks, special characters or a combination of these.
INPUT can only be terminated by a <RETURN>.

LENGTH contains a count of the characters accepted.
INPUT into the noun NOUN

NOUN is the noun you want the input data to be copied to.

SEE ALSO: INPUT LAYOUT/INPUT NUMBER/INPUT STRING
DISCUSSION

INPUT accepts: spaces, letters, digits, punctuation marks, special characters (for example *, &, %), or a combination of these.

The maximum number of characters accepted is 512.

Use INPUT LAYOUT if you want to limit the operator's input to a certain number of characters.

Every INPUT operation (as well as every $INPUT, CUT, JOIN, or RECEIVE) writes a new value to the target noun LENGTH. To preserve the contents of LENGTH, copy it to another noun before using INPUT, $INPUT, CUT, JOIN, or RECEIVE again.

The following editing keys are allowed:

INS	Toggles insert mode on/off.
DEL	Delete character above cursor and scroll characters.
BACKSPACE	Deletes character to the left of the cursor and scrolls the characters above and to the right of the cursor to the left by 1.
LEFT ARROW	Moves the cursor to the left by 1.
RIGHT ARROW	Moves the cursor to the left by 1.
HOME	Moves the cursor to the start of the value.
END	Moves the cursor to 1 past the end of the value.
CTRL-LEFT	Deletes everything to the left of the cursor and scrolls characters above and to the right of the cursor left to the home position.
CTRL-RIGHT	Deletes everything to the right of the cursor.
ESC	Restarts. Clears the current value and redisplays the

original value. The first character delete and replace action is disabled.

<RETURN> will terminate the input.

The <ESC> key is remappable and may be substituted by another key, thus freeing the <ESC> key for other duties.

EXAMPLE

INPUT into the noun CUSTOMER NUMBER

Operator input is accepted and written into the noun CUSTOMER NUMBER.

~~~~~~~~~~~~~~~~~~~~~~~~~

INPUT LAYOUT INL Standard Verb Input/Output Operation
Accepts input of a maximum number of designated characters from the keyboard and writes it in a noun you designate. If 0 is specified as the maximum length it accepts one key input from the keyboard without requirung a carriage return, without showing it on the screen and without waiting for it.

512 is the maximum number of characters that may be specified as a layout.
INPUT LAYOUT using layout NUMBER into the noun NOUN

NUMBER is the maximum number of characters that can be entered before the input terminates. Must be a positive integer less than 1969 or 0 for the special case.

SEE ALSO: INPUT/INPUT KEY/INPUT STRING
DISCUSSION

INPUT LAYOUT accepts a maximum number of designated characters from the keyboard.

The maximum number of characters accepted is 512.

INPUT LAYOUT accepts: spaces, letters, digits, punctuation marks, special characters (for example *, &, %), or a combination of these.

Every INPUT LAYOUT operation (as well as every INPUT, CUT, JOIN, or RECEIVE) writes a new value to the target noun LENGTH. To preserve the contents of LENGTH, copy it to another noun before using INPUT LAYOUT, INPUT, CUT, JOIN, or RECEIVE again.

The following editing keys are allowed:
~~~~~~~~~~~~~~~~~~~~~~~~~

INS	Toggles insert mode on/off.
DEL	Delete character above cursor and scroll characters.
BACKSPACE	Deletes character to the left of the cursor and scrolls the characters above and to the right of the cursor to the left by 1.
LEFT ARROW	Moves the cursor to the left by 1.
RIGHT ARROW	Moves the cursor to the left by 1.
HOME	Moves the cursor to the start of the value.
END	Moves the cursor to 1 past the end of the value.
CTRL-LEFT	Deletes everything to the left of the cursor and scrolls characters above and to the right of the cursor left to the home position.
CTRL-RIGHT	Deletes everything to the right of the cursor.
ESC	Restarts. Clears the current value and redisplays the original value. The first character delete and replace action is disabled.

<RETURN> will terminate the input.

The <ESC> key is remappable and may be substituted by another key, thus freeing the <ESC> key for other duties.

EXAMPLE

INPUT LAYOUT using layout 5 into the noun CUSTOMER NUMBER

Operator input, up to five characters long, is accepted and written into the noun CUSTOMER NUMBER.

INPUT LAYOUT 0

INPUT LAYOUT 0 is a special case of INPUT LAYOUT. It allows one key input without a terminating return. The input will not echo to the screen, nor will it wait for a key to be pressed. If no keyboard data is ready, the destination noun will receive a null value, and LENGTH will be 0.

Otherwise, the destination noun will receive the one character, and length will be 1. If an extended key is input, the noun will receive both bytes, and LENGTH will be 2. Extended keys are all the keys that do not display anything, such as the function keys and their ctrl and alt states.

In order to use INPUT 0 to input more than one character, you will have to use a loop. To test for a non alphanumeric key, you must compare the key against a value. There are actually hundreds of combinations like Crtl-Z, Alt-Q, etc. A utility in the on-line help lets you determine any key value.

If you use a loop, and no key is pressed, INPUT LAYOUT 0 puts a null value into the noun and a zero in LENGTH. In other words, the keyboard is continuously polled and the value that is read, even if it is zero, is copied to two nouns. This consumes a lot of CPU time and will slow down the whole system. INPUT LAYOUT 0 is best suited for cases when a single key MIGHT be pressed to interrupt a job or something similar. It is best used sparingly, and not in a tight loop. When the program DEMANDS a key to be pressed, like in a menu, it is better to use the standard verb INPUT KEY (INK).

EXAMPLE

```
VERB  called  EDIT

1     INPUT LAYOUT using layout 0 into the noun CHARACTER
2     IF the value CHARACTER "=" the value "__D"  (F10 key)
3       do
          SOME VERB
        else
4         IF the value CHARACTER "=" the value "__?"  (F5 key)
            do
5             SOME OTHER VERB
            else
            end
        end

6     etc...
```

~~~~~~~~~~~~~~~~~~~~~~~~~

INPUT NUMBER INN Standard Verb Input/Output Operation

Accepts number input from the keyboard and edits a noun whose value is expected to be a number or an expression convertible to a number. Certain keys are allowed for editing, while any other control character will terminate the input and its value stored into a specified noun.

LENGTH contains a count of the characters accepted.
INPUT NUMBER using layout NUMBER into the noun NOUN terminating key into the noun NOUN2

NUMBER is the maximum number of characters that can be edited and must be a number or a noun containing a number of not greater than 16. NOUN is the noun to be edited. NOUN2 is the noun to receive the terminating control character.
SEE ALSO: INPUT/INPUT LAYOUT/INPUT STRING
DISCUSSION

Layout must be a number or a noun containing a number of 16 or less, since only 14 digits can by entered in any number (plus sign and decimal point). You can give it a higher number and you will be able to input more than 14 digits, but the result will be a string, not a number.
~~~~~~~~~~~~~~~~~~~~~~~~~

The layout has two parts: field width and decimal places. The maximum number of digits in a number is 14. Up to 13 may be on the right side of the point and the rest to the left. There is always at least one digit to the left of the point. The number of decimal places must be 13 or less. The whole number (field width) part must be at least 2 greater than the decimal (number of places) part of the layout to allow room for a leading zero and decimal point.

A numeric layout without any specified decimal places will disallow entry of decimal places.

A layout of 9.3 will give a total field width of 9 places with 3 decimal places.

The value of the noun to be edited is expected to be a number or an expression convertible to a number. If this is not the case, the "invalid number" message will be displayed and the job abandoned.

The value will be displayed starting from the current cursor position and according to the layout. Numbers will have no commas.

If a numeric value is larger than the layout specifies, the first digit will be a number sign (#). You may not edit this quantity (i.e., no editing keys will be allowed, only overtype mode will be available).

The cursor will be positioned to the ones digit of the number. If the first character typed is not an editing key, the entire field is cleared to spaces and you can type a new number.

The following editing keys are allowed:

BACKSPACE	Deletes digit above the cursor and scrolls the digits to the left of the cursor to the right by 1 if to the left of the decimal, otherwise the decimal place digit to the left of the cursor becomes a zero and the cursor moves to the left by 1. If it gets to the tenth digit, it is disallowed.
LEFT ARROW	Moves the cursor to the left by 1.
RIGHT ARROW	Moves the cursor to the left by 1.
CTRL-LEFT	Turns the whole number part to zero and moves the cursor to ones digit. Removes the minus sign.
CTRL-RIGHT	Turns all decimal places to zero and moves the cursor to the tenth digit if decimal places are specified, otherwise it is disallowed.
ESC	Restarts. Clears the current value and redisplays the original value. The first number delete and replace action is disabled.

Any other control character will terminate the input.

The underline key is not allowed. Any disallowed key generates a beep.

The cursor movement functions do not stop on the point, but skip over it. When a decimal point is

entered, the cursor skips from the current position to the tenth digit, if decimal places are specified in the layout, or if no places are specified, it is a disallowed key . The minus key will cause a minus sign to be displayed before the leftmost digit if room exists within the field, otherwise it is a disallowed key. The plus key will remove this leading minus. The only other allowed keys are the digits. When a digit is entered to the left of the point, digits to the left of the cursor are scrolled left by 1, and the digit typed appears above the cursor. The cursor does not move.

Insert mode is essentially always enabled when it is to the left of the decimal. Digits entered to the right of the point overwrite the digit above the cursor and the cursor moves to the right by 1.

If this sounds cryptic to you, don't worry. It is much more difficult to explain than to do, so just try it out.

~~~~~~~~~~~~~~~~~~~~~~~~

INPUT STRING INS Standard Verb Input/Output Operation
Accepts alphanumeric input from the keyboard and edits a noun. All data is considered an expression. Numeric noun values are converted to an expression. Certain keys are allowed for editing, while any other control character terminates the input and its value is stored into a specified noun.

LENGTH contains a count of the characters accepted.
INPUT STRING using layout NUMBER into the noun NOUN terminating key into the noun NOUN2

NUMBER is the maximum number of characters that can be edited and must be a number or a noun containing a number greater than 512. NOUN is the noun to be edited. NOUN2 is the noun to receive the terminating control character.
SEE ALSO: INPUT/INPUT LAYOUT/INPUT NUMBER
DISCUSSION

INPUT STRING is not really an input, but rather an editing verb.

Layout must be a number or a noun containing a number greater than zero. Layout theoretically has no upper limit, but practical considerations will limit it to a manageable size.

The cursor will be positioned to the start of the noun string. If the first character typed is not an editing key, the entire field is cleared to spaces and you can type a new string.

The following editing keys are allowed:

INS	Toggles insert mode on/off.
DEL	Delete character above cursor and scroll characters.
BACKSPACE	Deletes character to the left of the cursor and scrolls the characters above and to the right of the cursor to the left by 1.
~~~~~~~~~~~~~~~~~~~~~~~~

LEFT ARROW	Moves the cursor to the left by 1.
RIGHT ARROW	Moves the cursor to the left by 1.
HOME	Moves the cursor to the start of the value.
END	Moves the cursor to 1 past the end of the value.
CTRL-LEFT	Deletes everything to the left of the cursor and scrolls characters above and to the right of the cursor left to the home position.
CTRL-RIGHT	Deletes everything to the right of the cursor.
ESC	Restarts. Clears the current value and redisplays the original value. The first character delete and replace action is disabled.

Any other control character will terminate the input.

The <ESC> key is remappable and may be substituted by another key, thus freeing the <ESC> key for other duties.

The noun that is to be edited must not contain any control characters, because the underline key (control character) is not allowed. Any disallowed key generates a beep.

All data is considered ASCII. Numeric noun values are converted to ASCII. Numeric data entered is not converted to a number. This feature is useful for leading zero numeric expression data.

~~~~~~~~~~~~~~~~~~~~~~~~~

INTEGER NT Standard TargetNoun
Used with the verb SPLIT.

Contains the integer result of the most recent SPLIT operation or the most recent value copied to it.

SPLIT the value of ITEM
DISPLAY the value INTEGER

ITEM can be a number or a noun containing a number.

SEE ALSO: SPLIT/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

IPC STATUS IS Standard TargetNoun
Contains the status of your last operating system access operation.

All the standard verbs that perform operating system operations write the result of the operation into the noun IPC STATUS.

DISPLAY the value IPC STATUS

SEE ALSO: DOS COPY/DOS DELETE/DOS CLOSE READ
The following is a list of the IPC STATUS messages that are supported.

The number following the message is the equivalent DOS error status number, which can be looked up in a DOS technical reference for more detailed explanations. Since most of the DOS statuses never happen while doing IPC things, they end up as "UNKNOWN DOS ERROR".

IPC STATUS Message	DOS Error Number
END OF FILE	-1
OK	0
INVALID FUNCTION	1
FILE NOT FOUND	2
PATH NOT FOUND	3
UNKNOWN DOS ERROR	4
UNKNOWN DOS ERROR	5
UNKNOWN DOS ERROR	6
NOT ENOUGH MEMORY	7
BAD FORMAT	8
INVALID HANDLE	9
UNKNOWN DOS ERROR	10
UNKNOWN DOS ERROR	11
NOT ENOUGH MEMORY	12
ACCESS DENIED	13
UNKNOWN DOS ERROR	14
UNKNOWN DOS ERROR	15
UNKNOWN DOS ERROR	16
OK	17
NOT SAME DEVICE	18
UNKNOWN DOS ERROR	19
UNKNOWN DOS ERROR	20
UNKNOWN DOS ERROR	21
INVALID DATA	22
UNKNOWN DOS ERROR	23
TOO MANY OPEN FILES	24
UNKNOWN DOS ERROR	25
UNKNOWN DOS ERROR	26
UNKNOWN DOS ERROR	27
DEVICE FULL	28
UNKNOWN DOS ERROR	29
UNKNOWN DOS ERROR	30
UNKNOWN DOS ERROR	31
UNKNOWN DOS ERROR	32

INVALID DATA	33
INVALID DATA	34
UNKNOWN DOS ERROR	35
INVALID ACCESS	36

All messages are in upper case only. Keep that in mind when comparing the messages in the program.

~~~~~~~~~~~~~~~~~~~~~~~~~

JOIN JN Standard Verb Miscellaneous Data Manipulation
Joins two items and writes the result to the target noun HEAD. The result is always an expression. The JOIN operation does not affect the contents of the items joined.

The noun HEAD contains the result of the most recent JOIN.
LENGTH contains a count of the number of characters in HEAD after the JOIN.
JOIN the value ITEM1 to the value ITEM2

ITEM1 and ITEM2 can be nouns, numbers or expressions. ITEM1 is placed to the left of ITEM2.

SEE ALSO: CUT/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

LABEL LB Standard Verb Execution Flow Control
Defines a label in a verb definition. It does not perform any action, but merely marks a line within a verb definition that is used as the target of a GO TO instruction. A GO TO instructions tells PRAGMA to branch to a specific LABEL instruction. PRAGMA then continues executing the verb at the instruction following the label. A particular LABEL name can be used only once in a verb, but the same name can be used again in other verbs.
LABEL this line with NAME

NAME can be a string of 1-31 characters. The characters can be entirely alphabetic, entirely numeric or a combination of letters, digits, special characters (such as @ or #) and spaces.

SEE ALSO: GO TO/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

LENGTH LT Standard TargetNoun
Contains a count of the characters accepted by the most recent INPUT, $INPUT, INPUT LAYOUT, INPUT STRING, INPUT NUMBER, RECEIVE or DOS READ operation.
Contains a count of the characters in HEAD after a CUT or JOIN operation.

DISPLAY the value LENGTH
~~~~~~~~~~~~~~~~~~~~~~~~~

SEE ALSO: INPUT/CUT/JOIN
~~~~~~~~~~~~~~~~~~~~~~~~

MESSAGE DELETE MD Standard Verb Pragma Fileserver Access Operation
This verb tells the workstation to prevent the last message you received from being restored to a PRAGMA Fileserver message queue.

The noun EXTERNAL ECHO contains the status message of the Fileserver access operation.

MESSAGE DELETE

SEE ALSO: MESSAGE SEND/MESSAGE RECEIVE/
~~~~~~~~~~~~~~~~~~~~~~~~~~

MESSAGE RECEIVE MR Standard Verb Pragma Fileserver Access Operation
This verb tells the workstation to bring the first message out of the PRAGMA Fileserver message queue.

The noun EXTERNAL ECHO contains the status message of the Fileserver access operation.

MESSAGE RECEIVE from queue FILE

FILE is the name of the file that contains the record.

SEE ALSO: MESSAGE SEND/MESSAGE DELETEE/
~~~~~~~~~~~~~~~~~~~~~~~~

MESSAGE SEND MS Standard Verb Pragma Fileserver Access Operation
This verb tells the workstation to place a message onto the end of a PRAGMA Fileserver message queue.

The noun EXTERNAL ECHO contains the status message of the Fileserver access operation.

MESSAGE SEND onto queue FILE

FILE is the name of the file that contains the record.
~~~~~~~~~~~~~~~~~~~~~~~~

SEE ALSO: MESSAGE RECEIVE/MESSAGE DELETEE/
~~~~~~~~~~~~~~~~~~~~~~~~~

MOVE CURSOR DOWN   MCD Standard Verb Screen Operation
This verb does relative cursor positioning. It moves the cursor down the number of lines specified. If the cursor reaches the bottom of the screen it will stay on the bottom line and the screen will scroll up.

MOVE CURSOR DOWN the amount NUMBER

NUMBER must be a number or a noun containing a number.

SEE ALSO: MOVE CURSOR UP/MOVE CURSOR LEFT/MOVE CURSOR RIGHT
~~~~~~~~~~~~~~~~~~~~~~~~~

MOVE CURSOR LEFT   MCL Standard Verb Screen Operation
This verb does relative cursor positioning. It moves the cursor left the number of spaces specified. If the cursor reaches the beginning of a line it will wrap to the end of the previous line.

MOVE CURSOR LEFT the amount NUMBER

NUMBER must be a number or a noun containing a number.

SEE ALSO: MOVE CURSOR UP/MOVE CURSOR DOWN/MOVE CURSOR RIGHT
~~~~~~~~~~~~~~~~~~~~~~~~~

MOVE CURSOR RIGHT   MCR Standard Verb Screen Operation
This verb does relative cursor positioning. It moves the cursor right the number of spaces specified. If the cursor reaches the end of a line it will wrap to the beginning of the next line.

MOVE CURSOR RIGHT the amount NUMBER

NUMBER must be a number or a noun containing a number.

SEE ALSO: MOVE CURSOR UP/MOVE CURSOR DOWN/MOVE CURSOR LEFT
~~~~~~~~~~~~~~~~~~~~~~~~~

MOVE CURSOR UP   MCU Standard Verb Screen Operation

This verb does relative cursor positioning. It moves the cursor up the number of lines specified. If the cursor reaches the top line of the screen it will wrap to the bottom line without scrolling the screen down.

MOVE CURSOR UP the amount NUMBER

NUMBER must be a number or a noun containing a number.

SEE ALSO: MOVE CURSOR DOWN/MOVE CURSOR LEFT/MOVE CURSOR RIGHT
~~~~~~~~~~~~~~~~~~~~~~~~~~

MULTIPLY * Standard Verb Arithmetic Operation
Multiplies two items and places the result into the target noun PRODUCT.
The operands in the multiplication can be numbers or nouns containing numbers.
To multiply a series of three or more items you have to perform more than one multiplication.

The result of the multiplication is placed into the target noun PRODUCT.
MULTIPLY the value NUMBER by the value ITEM

NUMBER and ITEM can be numbers or nouns that contain a number.
The multiplication does not alter the original value(s) of the item.

SEE ALSO: PRODUCT/ /
~~~~~~~~~~~~~~~~~~~~~~~~~~

OBJECT OB Standard Noun Noun
OBJECT is a standard noun that in PRAGMA 4 no longer has any special meaning and it is a noun like any other noun.

A program converted from PRAGMA 3 will translate the object name to OB1, to avoid confusion with the standard noun OBJECT.
To avoid confusion we recommend that you do not use this noun.

SEE ALSO: / /
~~~~~~~~~~~~~~~~~~~~~~~~~~

OUTPUT OT Standard Verb Input/Output Operation
Sends text and numbers to the screen, the printer or the device attached to the serial port. The output destination is chosen by first including in the verb definition a DISPLAY, PRINT or SEND.
~~~~~~~~~~~~~~~~~~~~~~~~~~

OUTPUT the value ITEM

ITEM is the noun or expresion you want to output. Expressions must be enclosed in quotation marks ("). Control characters must be entered as expressions.

SEE ALSO: OUTPUT LAYOUT/SEND/DISPLAY
~~~~~~~~~~~~~~~~~~~~~~~~~

OUTPUT LAYOUT   OTL Standard Verb Input/Output Operation
Sends formatted text and numbers to the screen, the printer or the device attached to the serial port. The output destination is chosen by first including in the verb definition a DISPLAY, PRINT or SEND.

OUTPUT LAYOUT using layout LAYOUT the value ITEM

ITEM is the noun or expression you want to output. Expressions must be enclosed in quotation marks ("). Control characters must be entered as expressions. LAYOUT is the specification indicating the format to output ITEM.

SEE ALSO: OUTPUT/SEND/DISPLAY
~~~~~~~~~~~~~~~~~~~~~~~~~

PRINT   PR   Standard Verb Input/Output Operation
Prints text and numbers on your printer.
The output is printed from left to right, starting in the first available column. Characters from successive PRINT instructions are printed directly one after the other on the same line.
Control characters format the PRINT output.

PRINT the value ITEM

ITEM is the noun or expression you want to print. Expressions must be enclosed in quotation marks ("). Control characters must be entered as expressions.

SEE ALSO: PRINT LAYOUT/ASSIGN PRINTER/
~~~~~~~~~~~~~~~~~~~~~~~~~

PRINT LAYOUT   PRL Standard Verb Input/Output Operation
Prints text and numbers on your printer formatted as a column of a particular width. Expressions are printed left-aligned in the column, numbers right aligned in the column, with or without decimal point. Any unused spaces in the column are filled with the space (blank) character.

PRINT LAYOUT using layout LAYOUT the value ITEM
~~~~~~~~~~~~~~~~~~~~~~~~~

LAYOUT is the layout specification indicating the format in which the noun,
number or expression named in ITEM is to be displayed. A layout of 20 will
establish a column of the width of 20 characters. To layout several columns
on a line, use several PRINT LAYOUT instructions.
SEE ALSO: PRINT/ /
You may also enter a layout like 0.2. This will cause the value to be output in a field width that is the exact size of the number, with commas added, etc.

~~~~~~~~~~~~~~~~~~~~~~~~~

PRODUCT   PD   Standard TargetNoun
Used with the standard verb MULTIPLY.
Contains the result of the most recent multiplication or the most recent value copied to it.

DISPLAY the value PRODUCT

SEE ALSO: MULTIPLY/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

PUT TIME   PT   Standard Verb Miscellaneous Data Manipulation
Copies the current date and time from the system clock into a specified noun.

PUT TIME into the noun NOUN

NOUN can be any noun into which a string of 20 digits and spaces is copied to.
The format is YYYY MM DD hh mm ss. The hour is given according to the 24-hour clock.

SEE ALSO: EXTERNAL TIME/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

QUOTIENT   QT   Standard TargetNoun
Used with the standard verb DIVIDE.
Contains the result of the most recent division or the most recent value copied to it.
QUOTIENT itself can be a divisor or dividend.

DISPLAY the value QUOTIENT
~~~~~~~~~~~~~~~~~~~~~~~~~

SEE ALSO: DIVIDE/ /
~~~~~~~~~~~~~~~~~~~~~~~~

RECEIVE RV Standard Verb Input/Output Operation
Accepts input from a device attached to the serial I/O port and writes it into the noun you specify. RECEIVE is terminated if the character specified in the PRAGMA.DES file to terminate RECEIVE is received, a null character is received or 2048 characters have been received. The parameters of the serial port must be specified in the operating system.
The target noun LENGTH contains a count of the number of characters received.
RECEIVE into the noun NOUN

NOUN can be any noun.

SEE ALSO: RECEIVE LAYOUT/RECEIVE FILE/SEND
~~~~~~~~~~~~~~~~~~~~~~~~~~

RECEIVE LAYOUT RVL Standard Verb Input/Output Operation
Accepts input from a device attached to the serial I/O port and writes it into the noun you specify. It functions like RECEIVE except that it lets you specify the maximum number of characters to accept or the number of seconds to wait in the receive loop before timing out. If no timeout is given the program will wait forever until the receive terminator character is seen.
The target noun LENGTH contains a count of the number of characters received.
RECEIVE using layout LAYOUT into the noun NOUN

LAYOUT is a specification indicating the maximum number of characters to accept when is is a positive integer from 1 to 2048 inclusive. The optional digit to the right of the decimal point is the number of seconds to wait before timing out. NOUN can be any noun.
SEE ALSO: RECEIVE /RECEIVE FILE/SEND LAYOUT
~~~~~~~~~~~~~~~~~~~~~~~~

REFERENCE RF Standard TargetNoun
Contains the REFERENCE of the most recently accessed record or the most recent value copied to it.

It is used with the file access operations and external file access operations verbs.

DISPLAY the value REFERENCE
~~~~~~~~~~~~~~~~~~~~~~~~

SEE ALSO: GET/EXTERNAL GET/SAVE
~~~~~~~~~~~~~~~~~~~~~~~~~

RELEASE RL Standard Verb File Access Operation
This verb tells the workstation to let go of the file record it currently controls.

Used only with mock internal (old) record locking scheme.

RELEASE

SEE ALSO: EXTERNAL RELEASE/ /
~~~~~~~~~~~~~~~~~~~~~~~~~~

REPEAT RP Standard Verb Direction Operation
This verb tells PRAGMA to go back to the first line of the current verb and begin again the execution of the verb.

REPEAT

SEE ALSO: GO TO/RETURN/RETURN TO START
~~~~~~~~~~~~~~~~~~~~~~~~~

RETURN RT Standard Verb Execution Flow Control
Halts the execution of the current verb and returns PRAGMA to the verb that called the current verb or to the START Message.

If the RETURN verb is omitted from a definition and the last line is not a GO TO, RETURN TO START or REPEAT, PRAGMA automatically returns to the prior verb. RETURN need never be the last line in a verb definition.
RETURN

SEE ALSO: RETURN TO START/RETURN TO OPERATING SYSTEM/REPEAT
~~~~~~~~~~~~~~~~~~~~~~~~~

RETURN TO START RS Standard Verb Execution Flow Control

Halts whatever is being executed and returns to the START message.

RETURN TO START

SEE ALSO: RETURN TO OPERATING SYSTEM/RETURN/REPEAT
~~~~~~~~~~~~~~~~~~~~~~~~~~

SAVE SV Standard Verb File Access Operation
Writes a new record and reference to the specified file.
The CHOICE is not important to be specified during this operation, since it has been defined in the file definition.
After a successful SAVE operation the reference for the saved record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.
SAVE reference CHOICE in file FILE

CHOICE can be anything since it is not used by this operation.
FILE is the name of the file where the record is written to.

SEE ALSO: SAVE THIS/BEGIN THIS/GET
~~~~~~~~~~~~~~~~~~~~~~~~~~

SAVE FIRST SVF Standard Verb File Access Operation
Updates the first record in the specified file.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

SAVE FIRST in file FILE

FILE is the name of the file whose FIRST record is being updated.

SEE ALSO: SAVE/ /
~~~~~~~~~~~~~~~~~~~~~~~~~~

SAVE LAST SVL Standard Verb File Access Operation
Updates the last record in the specified file.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.
~~~~~~~~~~~~~~~~~~~~~~~~~~

SAVE LAST in file FILE

FILE is the name of the file whose LAST record is being updated.

SEE ALSO: SAVE/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

SAVE NEXT SVN Standard Verb File Access Operation
Updates the next record in the specified file.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

SAVE NEXT in file FILE

FILE is the name of the file whose NEXT record is being updated.

SEE ALSO: SAVE/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

SAVE THIS SVT Standard Verb File Access Operation
Updates the present record in the specified file.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

SAVE THIS in file FILE

FILE is the name of the file where the record is being updated.

SEE ALSO: SAVE/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

SEND SD Standard Verb Input/Output Operation
Outputs data to the device attached to the serial I/O port. All the ASCII characters can be transmitted, the underline character only as part of a control sequence. The null character terminates the transmission sequence or you can specify a different character in the PRAGMA.DES file.
The parameters of the serial port must be specified in the operating system.

SEND the value ITEM
~~~~~~~~~~~~~~~~~~~~~~~~~

ITEM can be a noun or an expression.

SEE ALSO: SEND LAYOUT/SEND FILE/RECEIVE
~~~~~~~~~~~~~~~~~~~~~~~~~

SEND LAYOUT SDL Standard Verb Input/Output Operation
Outputs data to the device attached to the serial I/O port. It functions like SEND except that it lets you specify the length and/or format of the output data.

SEND using layout LAYOUT the value ITEM

LAYOUT is the layout specification indicating the format in which the noun or expression named in ITEM is to be sent.
ITEM can be a noun or an expression.

SEE ALSO: SEND/SEND FILE/RECEIVE LAYOUT
~~~~~~~~~~~~~~~~~~~~~~~~~~

SET BACKGROUND COLORSBC Standard Verb Screen Operation
This verb performs the color selection of the background of the screen.
As color you must use one of the standard nouns that describe a color.
The color selection remains in effect until a new color is set.
To paint the whole screen with a background color, set a color with this verb followed by the standard verb ERASE SCREEN. SET BACKGROUND COLOR must always be used after SET DISPLAY ATTRIBUTE. Has no effect on monochrome monitors.
SET BACKGROUND COLOR to the color COLOR NAME

COLOR NAME is one of the standard nouns that specify a color.

SEE ALSO: SET FOREGROUND COLOR/ERASE SCREEN/COLOR BLACK
~~~~~~~~~~~~~~~~~~~~~~~~~

SET CURSOR POSITION SCP Standard Verb Screen Operation
This verb positions the cursor anywhere on the screen. The desired location on the screen where you want the cursor to be must first be copied to the standard nouns CURSOR ROW (CR) and CURSOR COLUMN (CC).

COPY the value ROW to the noun CURSOR ROW
COPY the value COLUMN to the noun CURSOR COLUMN
SET CURSOR POSITION

ROW and COLUMN are numbers or nouns containing numbers that indicate the desired location on the screen.
~~~~~~~~~~~~~~~~~~~~~~~~~

SEE ALSO: CURSOR ROW/CURSOR COLUMN/
~~~~~~~~~~~~~~~~~~~~~~~~~

SET DISPLAY ATTRIBUTSDA Standard Verb Screen Operation
This verbs sets the display attributes of the screen. As attribute you must use one of the standard attribute nouns.
SET DISPLAY ATTRIBUTE must always be used before SET BACKGROUND COLOR or SET FOREGROUND COLOR. On color monitors only the ATTRIBUTE BLINK and ATTRIBUTE BOLD have an effect.
Be sure that you use the correct TERMINFO when using a monochrome monitor.
SET DISPLAY ATTRIBUTE to the attribute ATTRIBUTE NAME

ATTRIBUTE NAME is one of the standard attribute nouns.

SEE ALSO: ATTRIBUTE BLINK/ATTRIBUTE NORMAL/SET BACKGROUND COLOR
~~~~~~~~~~~~~~~~~~~~~~~~~

SET FOREGROUND COLORSFC Standard Verb Screen Operation
This verb performs the color selection of the foreground of the screen.
As color you must use one of the standard nouns that describe a color.
The color selection remains in effect until a new color is set.
SET FOREGROUND COLOR must always be used after SET DISPLAY ATTRIBUTE.
Has no effect on monochrome monitors.

SET FOREGROUND COLOR to the color COLOR NAME

COLOR NAME is one of the standard nouns that specify a color.

SEE ALSO: SET BACKGROUND COLOR/COLOR BLACK/SET DISPLAY ATTRIBUTE
~~~~~~~~~~~~~~~~~~~~~~~~~

SPLIT   SP   Standard Verb Arithmetic Operation
Breaks a number or a noun containing a number into its integer (whole number) and decimal (fraction) parts. The integer is written to the target noun INTEGER and the decimal is written to the target noun FRACTION.
The value of the original item is not affected by the SPLIT operation.

SPLIT the value of ITEM

ITEM can be a number or a noun containing a number.

SEE ALSO: INTEGER/FRACTION/MODULO
~~~~~~~~~~~~~~~~~~~~~~~~~

SUBTRACT   -   Standard Verb Arithmetic Operation

Subtracts one item from another and places the difference in the target noun DIFFERENCE.
The numbers or nouns used in the subtraction are not affected.

The result of the subtraction is placed into the target noun DIFFERENCE.

SUBTRACT the value of ITEM 1 from the value of ITEM 2

ITEM 1 and ITEM 2 can be numbers or nouns containing numbers.

SEE ALSO: DIFFERENCE/ADD/MULTIPLY
~~~~~~~~~~~~~~~~~~~~~~~~~~

SUM SM Standard TargetNoun
Used with the standard verb ADD.

Contains the result of the most recent addition or the most recent value copied to it.

DISPLAY the value SUM

SEE ALSO: ADD/ /
~~~~~~~~~~~~~~~~~~~~~~~~~~

TAIL TL Standard TargetNoun
Used with the standard verb CUT.

Contains the right hand result of the most recent CUT operation or the most recent value copied to it.

DISPLAY the value TAIL

SEE ALSO: HEAD/CUT/LENGTH
~~~~~~~~~~~~~~~~~~~~~~~~~~

VERB VB Standard InteraDefinition Operation
Begins the definition of a new verb.
If a verb is to be an object verb, you specify this at the beginning of a verb definition.
A verb definition is a program which consists of a series of instructions,
~~~~~~~~~~~~~~~~~~~~~~~~~~

each of which contains another verb. A definition can include standard verbs, verbs you define and undefined verbs. Interactive-only verbs cannot be used.
VERB called NAME

NAME can be a string of 1 to 31 characters. The characters can be entirely alphabetic or a combination of letters, digits, special characters and spaces. Leading, trailing and multiple contiguous spaces are not allowed. Careful when using special characters and spaces if you plan to compile!
SEE ALSO: ALTER/FORGET/
~~~~~~~~~~~~~~~~~~~~~~~~~

FILE FL Standard InteraDefinition Operation
Begins the definition of a new file. After naming the file you must type the names of each of the nouns that define the fields that the file's record will contain. Then you must create the empty PFM file in the operating system with the verb CREATE PFM FILE or EXTERNAL CREATE PFM FILE.
If the file has a mockflag of zero or is a Btrieve file you must define a .DES .DES file and use the appropriate utility.
FILE called NAME
The shortname is SHORTNAME

NAME can be a string of 1 to 31 characters. SHORTNAME is supplied by PRAGMA and is the NAME shortened to 8 characters and is the name of the file in the operating system. SHORTNAME can be edited if desired.
SEE ALSO: CREATE PFM FILE/EXTERNAL CREATE PFM FILE/VERB
~~~~~~~~~~~~~~~~~~~~~~~~~

ALTER AL Standard InteraDefinition Operation
Changes the definition of a verb or file by adding lines, removing lines or changing the contents of lines.
The following keys perform the following functions:
<RETURN> Gets you back to START. <TAB> Moves one item in EDIT mode.
<ARROW DOWN> <PAGE DOWN> moves you forward in the verb.
<ARROW UP> <PAGE UP> moves you backwards in the verb.
ALTER the definition of NAME

NAME is the verb or file you want to alter.

SEE ALSO: VERB/FILE/RENAME
~~~~~~~~~~~~~~~~~~~~~~~~~

FORGET FG Standard InteraDefinition Operation
Remove a verb, noun or file from the vocabulary.
When you type the name of an item you want to remove, PRAGMA searches the vocabulary to be certain that the name does not occur in any definition. If it is used PRAGMA displays a list of the verbs and or files where it occurs and does not remove it from the vocabulary.

FORGET what NAME
~~~~~~~~~~~~~~~~~~~~~~~~~

NAME is the name of the verb, noun or file you want to remove.

SEE ALSO: ALTER/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

RENAME RN Standard InteraGeneral Vocabulary Operation
Changes the name of a verb, file or noun currently in the vocabulary. All occurrencies of the original name in all file and verb definitions are changed.
Two reasons to change a name are to correct a misspelling or to use a name that is more meaningful.

RENAME the item NAME1 to be NAME2

NAME1 can be the name of a known file, verb or noun, although it need not be defined yet.
NAME2 must NOT be known to PRAGMA and must conform to the naming conventions.

SEE ALSO: EXCHANGE/EXCHANGE ALL/
~~~~~~~~~~~~~~~~~~~~~~~~~~

EXCHANGE XC Standard InteraOperating System Operation
Replaces an item occurring in a verb or file definition with another item.
Do not confuse the EXCHANGE operation with the RENAME operation. RENAME changes an item's name without affecting the item's contents. EXCHANGE swaps an entire item - its name AND contents - with another entire item.
Within certain limits you can also EXCHANGE standard verbs.

EXCHANGE occurrencies of ITEM1 with ITEM2 inside the definition of VERB - FILE

ITEM1 and ITEM2 can be numbers, nouns, expressions, files or user defined verbs, including object verbs or standard words. Both items must be of the same type. Both items must already be in the vocabulary.
VERB or FILE is the verb or file whose definition you wish to change.
SEE ALSO: EXCHANGE/EXCHANGE ALL/
EXCHANGE (XC)

EXCHANGE replaces an item occurring in a verb or file definition with another item. Do not confuse the EXCHANGE operation with the RENAME operation. RENAME changes an item's name without affecting the item's contents. EXCHANGE swaps an entire item - its name AND contents - with another entire item.

To be able to exchange item1 with item2, item2 must be the same type as item1. For example, if item1 is a noun, item2 must also be a noun. Or if a verb has two objects, it can only be exchanged with a verb that also has two objects.

EXCHANGE is an interactive use only verb, that is, it can only be used from the START prompt and not from within a program.

After having specified the item to be exchanged and the item you want to exchange it with, PRAGMA checks if the items can be exchanged. If the exchange can take place, PRAGMA will prompt you for the name of the verb (or file if applicable) inside which you want to perform the exchange. After having done the exchange, PRAGMA will inform you of the number of times the exchange was performed. After having done an exchange you are given the possibility to do other exchanges with the same items. Simply hit <RETURN> to terminate the exchange procedure.

EXAMPLES

The following example illustrates how you can use EXCHANGE to correct a misspelling. When the file shown below was originally defined, the noun ADDRESS was misspelled ADRESS.

```
FILE  called  CUSTOMER

1  CUSTOMER INDEX SELECTION NOUN
2  NAME
3  ADRESS
4  ZIP
5  etc.
```

This error wasn't discovered until you attempted to include the noun ADDRESS in the definition of another verb. At that point PRAGMA responded with a message indicating that ADDRESS was not known and if you wanted to add it to the vocabulary as a new noun. Because of the different spelling, PRAGMA treated ADDRESS as a different noun. You went ahead with the definition of the new noun ADDRESS, deciding to correct the situation later.

You cannot correct this situation with a RENAME operation because both names now exist in PRAGMA's vocabulary. You may use the following EXCHANGE instruction instead:

```
EXCHANGE  occurrences of  ADRESS   with  ADDRESS

  inside the definition of  CUSTOMER
```

This replaces ADRESS with ADDRESS. Then you can use the verb FORGET to remove ADRESS from PRAGMA's vocabulary. Of course you could have also used ALTER to edit the file definition of CUSTOMER and replaced the misspelled noun. But should you have to exchange multiple occurrences of an item, EXCHANGE will come in handy.

Sometimes you may not want to alter the definition of a verb that is used by other verbs without first testing the changes.

EXCHANGE helps you do this. You create a copy of the verb, make the changes, and then test the verb. When you are satisfied that it works, you exchange the old verb with the new.

~~~~~~~~~~~~~~~~~~~~~~~~

EXCHANGE ALL XA Standard InteraOperating System Operation
It is the equivalent of EXCHANGE except that it replaces all occurrences of an item with another item in all verb and file definitions in the vocabulary. You can select the range (or all) you want to exchange.

Use this verb with care. The changes it makes are global, and if not thought out properly could cause very serious problems, which are difficult to undo.
EXCHANGE ALL occurrencies of ITEM1 with ITEM2

ITEM1 and ITEM2 can be numbers, nouns, expressions, files or user defined verbs, including object verbs or standard words. Both items must be of the same type. Both items must already be in the vocabulary.

SEE ALSO: EXCHANGE/RENAME/
EXCHANGE ALL (XA)

EXCHANGE ALL is the equivalent of EXCHANGE (XC) except that it replaces all occurrences of an item with another item in all verb and file difinitions in the vocabulary. Please consult the reference of EXCHANGE for the instructions on how to exchange an item.

When using EXCHANGE ALL, you can select the range of verbs or files you want to exchange. Or you can select all of them.

Use this verb with care. The changes it makes are global, and if not thought out properly could cause very serious problems. It is best to make a backup of your vocabulary before using this verb.

For example, if you exchange all occurrences of an item with another item that already occurs in the system, it will probably be very difficult to undo the exchange. Suppose there were occurrences of 1's and 2's before an exchange and you exchange all the 1's with 2's. You realize then that you wanted to retain some of the 1's. You will not be able to restore the 1's using EXCHANGE ALL without affecting all of the 2's. This is illustrated below.

```
Before the EXCHANGE ALL:     1 1 2 2
EXCHANGE ALL 1's with 2's:   2 2 2 2
EXCHANGE ALL 2's with 1's:   1 1 1 1
```

The original 1's are restored, but the original 2's are now 1's.

~~~~~~~~~~~~~~~~~~~~~~~~

LIST VERBS LV Standard InteraListing and Report Operation
Displays an alphabetized list of all verbs currently in the vocabulary.
By default the listing is displayed on the screen, but it can be redirected

to the printer or the serial port by typing a redirection key immediately
before entering the verb. The default redirection keys are <F5> screen, <F6>
printer and <F7> communications. These keys can be changed in PRAGMA.DES.
Before listing you can select the range (or all) you want to list.
LIST VERBS
Enter the name of the first item to include in the list (NULL for lowest) :
Enter the name of the last item to include in the list (NULL for highest) :

NULL is the <RETURN> key.

SEE ALSO: LIST STANDARDS/LIST FILES/LIST NOUNS
~~~~~~~~~~~~~~~~~~~~~~~~~

LIST FILES   LF   Standard InteraListing and Report Operation
Displays an alphabetized list of all files currently in the vocabulary.
By default the listing is displayed on the screen, but it can be redirected
to the printer or the serial port by typing a redirection key immediately
before entering the verb. The default redirection keys are <F5> screen, <F6>
printer and <F7> communications. These keys can be changed in PRAGMA.DES.
Before listing you can select the range (or all) you want to list.
LIST FILES
Enter the name of the first item to include in the list (NULL for lowest) :
Enter the name of the last item to include in the list (NULL for highest) :

NULL is the <RETURN> key.

SEE ALSO: LIST REFERENCES/EXTERNAL LIST REFERENCES/
~~~~~~~~~~~~~~~~~~~~~~~~~

LIST NOUNS   LN   Standard InteraListing and Report Operation
Displays an alphabetized list of all nouns currently in the vocabulary.
By default the listing is displayed on the screen, but it can be redirected
to the printer or the serial port by typing a redirection key immediately
before entering the verb. The default redirection keys are <F5> screen, <F6>
printer and <F7> communications. These keys can be changed in PRAGMA.DES.
Before listing you can select the range (or all) you want to list.
LIST NOUNS
Enter the name of the first item to include in the list (NULL for lowest) :
Enter the name of the last item to include in the list (NULL for highest) :

NULL is the <RETURN> key.

SEE ALSO: LIST VERBS/LIST FILES/LIST UNDEFINEDS
~~~~~~~~~~~~~~~~~~~~~~~~~

LIST REFERENCES   LR   Standard InteraListing and Report Operation
Displays an alphabetized and numbered list of the references to the records
in the named file, using INTERNAL file manager. You may choose the key by
which the references are listed.
References that are expressions are shown in quotes. Before listing you can
select the range (or all) you want to list. When selecting a range you must
enclose expressions in quotes, otherwise a number is expected.
~~~~~~~~~~~~~~~~~~~~~~~~~

LIST REFERENCES in file FILE using key number NUMBER

FILE is the name of the file whose references are to be listed.
NUMBER is the number of the key by which you want to select the references.

SEE ALSO: EXTERNAL LIST REFERENCES/LIST VERBS/
The listing of LIST REFERENCES (LR) or EXTERNAL LIST REFERENCES (XLR) can be redirected to the printer or the serial port by typing a redirection key immediately before entering the LR or XLR command.

The default redirection keys are:

<F5> for the screen
<F6> for the printer
<F7> for the communications port

These keys can be changed by making an appropriate entry into the description file PRAGMA.DES.

Before the listing starts you will be asked for the upper and lower bound of the listing. You must type either a numeric value or an expression. The input routine demands that expressions be enclosed in quotes. If an opening quote if not typed, a number is expected, and ONLY digits, the minus sign, or the decimal point may be typed. When a number is expected, all other characters are beeped. Once a quote is typed ("), the routine switches into expression mode, and any value may be typed, including control characters.

A number entered in numeric mode is not validated as entered, but is validated after return is pressed. An invalid numeric found at this point will cause a beep, it will be erased, and you must then reenter it.

~~~~~~~~~~~~~~~~~~~~~~~~~

LIST UNDEFINEDS LU Standard InteraListing and Report Operation
Displays an alphabetized list of all undefined verbs and files in the vocab. By default the listing is displayed on the screen, but it can be redirected to the printer or the serial port by typing a redirection key immediately before entering the verb. The default redirection keys are <F5> screen, <F6> printer and <F7> communications. These keys can be changed in PRAGMA.DES. Before listing you can select the range (or all) you want to list.
LIST UNDEFINEDS
Enter the name of the first item to include in the list (NULL for lowest) :
Enter the name of the last item to include in the list (NULL for highest) :

NULL is the <RETURN> key.

SEE ALSO: LIST VERBS/LIST NOUNS/LIST FILES
~~~~~~~~~~~~~~~~~~~~~~~~~

TRACE TR Standard InteraListing and Report Operation
Displays an alphabetized list of every verb and/or file in which a specified

item appears, together with the number of time it is used in an item and
globally.
The listing can be redirected to the printer or serial port. See LIST VERBS.
Before listing you can select the items (or all of them) you want to search.

TRACE occurrences of ITEM
Enter the name of the first item to include in the search (NULL for lowest) :
Enter the name of the last item to include in the search (NULL for highest) :

ITEM can be a noun, number, expression, verb or file.

SEE ALSO: LIST VERBS/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

STATUS   SS   Standard InteraListing and Report Operation

STATUS

SEE ALSO: / /
~~~~~~~~~~~~~~~~~~~~~~~~

RECAP   RC   Standard InteraListing and Report Operation
Lists the entire definition of the specified verb or file.
By default the listing is displayed on the screen, but it can be redirected
to the printer or the serial port by typing a redirection key immediately
before entering the verb. The default redirection keys are <F5> screen, <F6>
printer and <F7> communications. These keys can be changed in PRAGMA.DES.

RECAP the definition of ITEM

ITEM can be a verb or file name.

SEE ALSO: RECAP ALL/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

RECAP ALL   RA   Standard InteraListing and Report Operation
Lists the definition of every verb and file in the vocabulary
By default the listing is displayed on the screen, but it can be redirected
to the printer or the serial port by typing a redirection key immediately
before entering the verb. The default redirection keys are <F5> screen, <F6>
printer and <F7> communications. These keys can be changed in PRAGMA.DES.
~~~~~~~~~~~~~~~~~~~~~~~~~

Before listing you can select the range (or all) you want to list.
RECAP ALL
Enter the name of the first item to include in the list (NULL for lowest) :
Enter the name of the last item to include in the list (NULL for highest) :

NULL is the <RETURN> key.

SEE ALSO: RECAP/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

LIST STANDARDS   LS   Standard InteraListing and Report Operation
Displays an alphabetized list of all the standard words in the vocabulary.
By default the listing is displayed on the screen, but it can be redirected
to the printer or the serial port by typing a redirection key immediately
before entering the verb. The default redirection keys are <F5> screen, <F6>
printer and <F7> communications. These keys can be changed in PRAGMA.DES.
Before listing you can select the range (or all) you want to list.
LIST STANDARDS
Enter the name of the first item to include in the list (NULL for lowest) :
Enter the name of the last item to include in the list (NULL for highest) :

NULL is the <RETURN> key.

SEE ALSO: LIST VERBS/LIST FILES/LIST NOUNS
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL LIST REFEREXLR Standard InteraListing and Report Operation
Displays an alphabetized and numbered list of the references to the records
in the named file, using EXTERNAL file manager. You may choose the key by
which the references are listed.
References that are expressions are shown in quotes. Before listing you can
select the range (or all) you want to list. When selecting a range you must
enclose expressions in quotes, otherwise a number is expected.
EXTERNAL LIST REFERENCES in file FILE using key number NUMBER

FILE is the name of the file whose references are to be listed.
NUMBER is the number of the key by which you want to select the references.

SEE ALSO: LIST REFERENCES/LIST VERBS/
The listing of LIST REFERENCES (LR) or EXTERNAL LIST REFERENCES (XLR) can be redirected to the printer or the serial port by typing a redirection key immediately before entering the LR or XLR command.

The default redirection keys are:

- <F5> for the screen
- <F6> for the printer
- <F7> for the communications port

These keys can be changed by making an appropriate entry into the description file PRAGMA.DES.

Before the listing starts you will be asked for the upper and lower bound of the listing. You must type either a numeric value or an expression. The input routine demands that expressions be enclosed in quotes. If an opening quote if not typed, a number is expected, and ONLY digits, the minus sign, or the decimal point may be typed. When a number is expected, all other characters are beeped. Once a quote is typed ("), the routine switches into expression mode, and any value may be typed, including control characters.

A number entered in numeric mode is not validated as entered, but is validated after return is pressed. An invalid numeric found at this point will cause a beep, it will be erased, and you must then reenter it.

~~~~~~~~~~~~~~~~~~~~~~~~

INPUT KEY INK Standard Verb Input/Output Operation
Waits for a key to be pressed and does not do anything while waiting.
It is recommended to be used when the program demands a key to be pressed to go on, like in menus.
If no data is ready from the keyboard, INPUT KEY will stop executing until data is ready, therefore not wasting CPU time in a multiuser system.
LENGTH contains 1 for any regular, 2 for extended key pressed.
INPUT KEY into the noun NOUN

NOUN is the noun you want the key value to be written to.

SEE ALSO: INPUT LAYOUT/INPUT/INPUT NUMBER
DISCUSSION

INPUT KEY allows one key input without a terminating return. It will wait for a key to be pressed and the inputted key will not echo to the screen. If no keyboard data is ready, the program will not complete its work until a key has been pressed.

When the key has been pressed, the destination noun will receive the one character, and length will be 1. If an extended key is input, the noun will receive both bytes, and LENGTH will be 2. Extended keys are all the keys that do not display anything, such as function keys and their CRTL and ALT states.

INPUT KEY should always be used when a program demands a key to be pressed, like in a menu. INPUT KEY will minimize CPU time while waiting for the key to be pressed. This is especially important in a multiuser system like UNIX, where it is important to maximize system throughput and INPUT LAYOUT 0 should be used as little as possible.

The return codes of the extended keys can be determined in the utilities section of the online help. Be aware that the program displays control characters as a caret (^), and not as an underline, like you will have to write when using INPUT KEY.
~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~

ASSIGN PRINTER AP Standard Verb Input/Output Operation
Used to redirect the output to the printer. Upon startup the printer assignement of the PRAGMA.DES file is used. With ASSIGN PRINTER this selection can be overwritten. You can redirect the printer output to a valid printer port or to a file. If the file already exists data will be appended to it. If the file does not exist it will be created. You cannot route data to a print spooler. To do so you must use a SYSTEM CALL.
ASSIGN PRINTER to PRINTER PORT (or FILENAME)

PRINTER PORT must be a string or a noun containing the name of the printer port. Under DOS "LPT1", under SCO UNIX "/dev/lp0", for instance.
FILENAME must be a string or a noun containing a valid filename.

SEE ALSO: SYSTEM CALL/ASSIGN COMMUNICATIONS/PRINT
ASSIGN PRINTER and ASSIGN COMMUNICATIONS allow redirection of input/output. The single parameter to these words can be any constant (numeric or string), or any noun. This is expected to be the name of the file or port which input/output will be read/written from/to. For instance, under DOS, "LPT1" could be used with ASSIGN PRINTER to cause all subsequent printing to go to the port LPT1. Under Xenix, "/dev/lp0" would cause the same to occur. Alternately, a filename may be given to cause all data to be read/written there. Whenever AP or AC is used, it first closes the current file/port, if any, and then opens the new one (if it IS a file) for append. If you want to instead start with an empty file, you may first use DOS DELETE to remove the old file if it exists.

When the system starts up, it is possible to set an initial default printer and/or communications port/file via the PRAGMA.DES file. If no such selection is made, output to printer and/or comm port will be thrown away until AP and/or AC assigns a port or file. It is also possible to select this condition by using ASSIGN PRINTER and ASSIGN COMMUNICATIONS. If you ASSIGN PRINTER to "", all subsequent printed data will be discarded.

ASSIGN PRINTER does not route any data to a printer spooler. You may print directly to a printer, if you know the port it is attached to; or you can print into a file. If you then want the printer spooler to put that file in line to be printed, you must use a SYSTEM CALL to invoke the printer spooler. The details of this SYSTEM CALL will vary from operating system to operating system. Under Xenix, the "lp" command could be used. PRAGMA automatically closes the printer and communications files prior to a SYSTEM CALL, so this is not necessary. It also reopens them when the SYSTEM CALL returns. If you wish to remove a print spool file after sending it to the spooler via a SYSTEM CALL, you must close the file (by issuing a ASSIGN PRINTER to ""), and then DOS DELETE it. However, some spoolers may get mad if you do this before they have finished printing it, so be careful. Some operating systems may have spooling built in. They may trap any output to specific ports and spool it. In this case, a SYSTEM CALL may not be necessary.

ASSIGN COMMUNICATIONS is used for the verbs SEND, RECEIVE, and all of the file commands used with the Pragma Fileserver. SEND and RECEIVE both use the same file.

~~~~~~~~~~~~~~~~~~~~~~~~

ASSIGN COMMUNICATIONAC Standard Verb Input/Output Operation

Used to redirect the input/output stream to a communications port. On start-up the communications port assignement of the PRAGMA.DES file is used. With ASSIGN COMMUNICATIONS this selection can be overwritten. You can redirect the input/output stream to a valid communications port or a file. If the file already exists data will be appended to it. If the file does not exist it will be created. You cannot route data to a print spooler.
ASSIGN COMMUNICATIONS to COMMUNICATIONS PORT (or FILENAME)

COMMUNICATIONS PORT must be a string or a noun containing the name of the communications port. Under DOS and OS/2 "COM1", for instance.
FILENAME must be a string or a noun containing a valid filename.

SEE ALSO: SEND/RECEIVE/ASSIGN PRINTER
ASSIGN PRINTER and ASSIGN COMMUNICATIONS allow redirection of input/output. The single parameter to these words can be any constant (numeric or string), or any noun. This is expected to be the name of the file or port which input/output will be read/written from/to. For instance, under DOS, "LPT1" could be used with ASSIGN PRINTER to cause all subsequent printing to go to the port LPT1. Under Xenix, "/dev/lp0" would cause the same to occur. Alternately, a filename may be given to cause all data to be read/written there. Whenever AP or AC is used, it first closes the current file/port, if any, and then opens the new one (if it IS a file) for append. If you want to instead start with an empty file, you may first use DOS DELETE to remove the old file if it exists.

When the system starts up, it is possible to set an initial default printer and/or communications port/file via the PRAGMA.DES file. If no such selection is made, output to printer and/or comm port will be thrown away until AP and/or AC assigns a port or file. It is also possible to select this condition by using ASSIGN PRINTER and ASSIGN COMMUNICATIONS. If you ASSIGN PRINTER to "", all subsequent printed data will be discarded.

ASSIGN PRINTER does not route any data to a printer spooler. You may print directly to a printer, if you know the port it is attached to; or you can print into a file. If you then want the printer spooler to put that file in line to be printed, you must use a SYSTEM CALL to invoke the printer spooler. The details of this SYSTEM CALL will vary from operating system to operating system. Under Xenix, the "lp" command could be used. PRAGMA automatically closes the printer and communications files prior to a SYSTEM CALL, so this is not necessary. It also reopens them when the SYSTEM CALL returns. If you wish to remove a print spool file after sending it to the spooler via a SYSTEM CALL, you must close the file (by issuing a ASSIGN PRINTER to ""), and then DOS DELETE it. However, some spoolers may get mad if you do this before they have finished printing it, so be careful. Some operating systems may have spooling built in. They may trap any output to specific ports and spool it. In this case, a SYSTEM CALL may not be necessary.

ASSIGN COMMUNICATIONS is used for the verbs SEND, RECEIVE, and all of the file commands used with the Pragma Fileserver. SEND and RECEIVE both use the same file.

In DOS the baud rate, parity, data and stop bits for the COM port must be set with the DOS command MODE.

Example:

Under DOS, to set the communication port COM1 to a baud rate of 2400, no parity, 8 data bits and 1 stop bit you enter the following command at the DOS prompt:

mode COM1:24,n,8,1

~~~~~~~~~~~~~~~~~~~~~~~~

SYSTEM CALL SC Standard Verb Operating System Operation
Used to perform any operating system command from within PRAGMA.
When the command has finished executing, PRAGMA will resume where it left
off in the current verb.
Prior to executing a SYSTEM CALL command, all open files are closed. Upon
resumption, all files are reopened. Careful. During the execution of a
SYSTEM CALL the cursor position and screen attributes may get lost.
SYSTEM CALL command OPERATING SYSTEM COMMAND

OPERATING SYSTEM COMMAND is a command for the operating system to execute.
It has the same syntax as if you had typed it from the command line prompt.

SEE ALSO: CLOSE ALL FILES//
SYSTEM CALL (SC)

SYSTEM CALL lets you perform any operating system command from within PRAGMA. When the command has finished executing, PRAGMA will resume where it left off in the current verb.

SYSTEM CALL takes one parameter. This parameter is expected to be a command for the operating system to execute, just as if you had typed the same command from the command line prompt.

Prior to executing the system call command, PRAGMA closes all open files. This includes all user PFM files, all Btrieve files, PRVOCAB.PFM, PRMSG.PFM and PRSTD.PFM. The printer and communications files/ports are also closed. Upon resumption of the verb after the system call is complete, PRAGMA reopens PRVOCAB.PFM, PRMSG.PFM, PRSTD.PFM, and the printer and communications files/ports. The other PFM and Btrieve files in use will not be reopened unless they are used again.

Be aware that when closing all files the THIS pointer of all the files will be lost. For instance, you cannot get a record of a file, do a system call and then save the modified record. PRAGMA will have lost the information of which record you wanted to update. Yet it is essential that PRAGMA closes all files before doing a system call. The program that you may invoke with the system call may crash and the closed PRAGMA files ensure that they will not be corrupted.

It is also important to realize that when performing work outside of PRAGMA's control, as a system call does, PRAGMA may lose track of the screen (cursor position, attributes, colors) if the system call command changes them, or displays data to the screen. The SYSTEM CALL verb will perform a read of the cursor position (which does NOT affect the nouns CURSOR ROW and CURSOR COLUMN) when it returns from the system call so that PRAGMA may know if the cursor has been moved. However, this depends upon whether the screen being used has the ability to report its cursor position, and this may not always be the case. For this reason the best course of action after a system call is to
~~~~~~~~~~~~~~~~~~~~~~~~

set the attribute, colors and cursor position again.

Under DOS, where the available memory is a scarce commodity, you may not always be able to run large programs with a system call. DOS will simply refuse to load the program with a not enough memory message, exit, and you will be back in PRAGMA. The DOS error message will have flashed by very quickly and you will be left wondering what has happened. If you suspect that this is happening to you, run the system call example shown below to see how much memory you have left.

When doing a system call under DOS, the 386 version of PRAGMA keeps only very little memory for itself, about 40K. Sometimes in DOS you will need to free as much lower memory as possible when making a system call. To do this place as much of DOS and other programs (e.g., network programs) as you can into upper memory. Version 5 of DOS gives you this option. If your system calls do not use much lower memory, it is better not to place DOS or other programs high, but to leave the extended memory free for PRAGMA. Luckily these concerns do not apply for UNIX or OS/2.

The status of the system call operation is placed into the noun EXTERNAL ECHO. This status is a number; 0 means all is OK. However, this is also an operating system dependent issue. DOS, for instance, always returns a zero status, even if the command does not execute. A non zero status is only returned from DOS if it cannot find COMMAND.COM.

Whenever PRAGMA does a system call, the lines System Call Started and System Call ended are added to the PRAGMA.LOG file. Since a system call closes all PRAGMA files, it is the equivalent of an orderly shutdown. The log entry reflects this and will let you know if a crash occurs while doing a system call.

EXAMPLES

To check the amount of free memory under DOS you can run the following verb:

VERB called TEST SYSTEM CALL

1 SYSTEM CALL command "MEM"
2 INPUT KEY into the noun KEY

The input in line 2 is necessary to stop PRAGMA from going back to the START prompt and erasing the display of the available memory.

You must always be aware that most system calls are operating system dependent. So if you want to list all the files that are in the directory where PRAGMA is running, and you want to be able to do this for every operating system PRAGMA supports, you would do something like this:

VERB called DISPLAY DIRECTORY

```
1  IF the value OPERATING SYSTEM NAME "=" the value "UNIX"
      do
2        COPY the value "l" to the noun LIST COMMAND
      else
3        COPY the value "DIR" to the noun LIST COMMAND
      end
4  SYSTEM CALL command LIST COMMAND
5  INPUT KEY into the noun KEY
```

Luckily the listing command "DIR" works both for DOS and OS/2, otherwise we would have had to also test whether the operating system was DOS or OS/2. The example also shows that the command for the system call can, of course, also be copied to a noun.

~~~~~~~~~~~~~~~~~~~~~~~~~

```
SELECTIVE SAVE      SSV  Standard InteraGeneral Vocabulary Operation
Lets you mark one or more verbs in the vocabulary.  SELECTIVE SAVE then
builds a tree of all the verbs used by the marked verbs and also marks these
verbs.  You may view which verbs are marked and which not.
When all the verbs have been marked you may then order PRAGMA to discard all
the unmarked verbs and unused nouns.  File definitions are also discarded but
actual data files are not.
SELECTIVE SAVE  ...
Type name of item to save,
LV, LF, or LN to list items,
or RETURN if no more.       -> VERBNAME
VERBNAME is the name of the verb from which the tree of marked verbs will be
built.
SEE ALSO: LIST VERBS/LIST FILES/LIST NOUNS
```

SELECTIVE SAVE does not attempt to discard file data. File definitions may be discarded, but the actual data in those files will not. PRAGMA does not attempt to find out which file manager the discarded files are used with, or even if the .PFM or .BTR files are present. It is up to you to delete the file manager data files if you wish to. There is also no more option to mark all files as saved, as it is unnecessary. SELECTIVE SAVE is now only a vocabulary operation.

The files and vocabulary have effectively been separated in PRAGMA. File definitions are considered to be vocabulary, whereas the file records themselves, and the files they live in, are not.

SELECTIVE SAVE does not compact your PRVOCAB.PFM file.

~~~~~~~~~~~~~~~~~~~~~~~~~

```
SAVE SCREEN IMAGE   SSI  Standard Verb  Screen Operation
Saves into a noun a complete screenful of data.
All the text, attributes of color of the text, the currently selected attri-
```

butes and colors and the cursor position are also saved.
The resulting noun can be DISPLAYed to restore the image to what it was at
the moment SAVE SCREEN IMAGE was called. BEWARE! SAVE SCREEN IMAGE will NOT
save screens painted by programs other than PRAGMA.
SAVE SCREEN IMAGE into the noun NOUN

NOUN must be a noun and will contain all the information to restore the
screen.

SEE ALSO: DISPLAY/SAVE SCREEN WINDOW IMAGE/SAVE SCREEN WINDOW TEXT
SAVE SCREEN IMAGE will put into the destination noun a complete screenfull of data. All of the text of the screen is saved AND all of the attributes and colors of the text are saved as well; plus the currently selected attributes and colors, and the cursor position.

The resulting noun can then be simply DISPLAYed to restore the image to what it was at the moment SAVE SCREEN IMAGE took this photograph. The cursor will be returned to its position at the time the picture was taken. The character attributes and colors that were currently selected at the time the image was saved are also restored (as they may have been different than any of the attributes or colors on the screen at the moment).

~~~~~~~~~~~~~~~~~~~~~~~~

TRANSLATE TO C   TRC Standard InteraGeneral Vocabulary Operation
Marks a tree or trees of items and then translates the marked items to
the C language. The resulting C code can be then compiled with an appropriate
C compiler to an executable file.

TRANSLATE TO C
Type noun of item to translate,
LV, LF, or LN to list items
or RETURN if no more.

SEE ALSO: SELECTIVE SAVE/ /
TRANSLATE TO C operates similarly to SELECTIVE SAVE, in that it marks a tree or trees of items, and only those items marked are translated.

You do not have to selective first the items you want to translate.

TRANSLATE TO C will ask repeatedly for items to mark. For each item entered, its complete tree will be marked (all items referenced by it or any of the verbs it uses). In the case of building a library, it may be necessary to choose multiple items if the library is intended to contain a number of different groups of functions. In the case of building a program, it will usually be the case that you simply mark everything from AUTOEXEC, and nothing more.

Either when building a program or a library, it will sometimes be desired to skip over certain groups of verbs if those verbs were previously translated and or compiled and made into a library.
~~~~~~~~~~~~~~~~~~~~~~~~

In order to do this, after typing items to mark, TRC asks for items to unmark. It is intended that you type the names of the groups of programs that are already in libraries that you wish to skip from this translation pass. All items typed here will be unmarked, for their full tree, just like the marking process in reverse. At the end of this process you will have remaining as marked only those programs which need to be translated to C.

The next prompt asks for the name of the destination file. Type the name of the executable file or library, but without any extension.

The next prompt asks whether this file will be a program or library.

A prompt next asks for the path to the file HIO.DEF. This file is only needed when linking for OS/2. If you do not intend to use OS/2, simply press RETURN here. Otherwise, enter the correct path. Do not enter a backslash at the end of the path.

The next prompt asks for a list of the libraries to use when linking this program. Of course you must answer this correctly, in accordance with what you have unmarked above, or linking will not complete. You must NOT use wildcard characters here. Each library must be specified individually. Path names may be added if desired.

Following this, the process begins. The only files created are .C files and the necessary files for linking, which all begin with the name given for the destination file with appropriate extensions.

~~~~~~~~~~~~~~~~~~~~~~~~~

DEBUG DB Debugger Verb Debugging Operation
Enters the debugger from the START Message.
You get a clear screen with the debugger prompt (similar to the START message, except it says "DEBUG").
To exit the debugger and return to START, simply press <RETURN> at the debug prompt without typing a command.
For more information, consult the "Debugger" section of the online help.
DEBUG

SEE ALSO: LOAD VERB/RUN VERB/
~~~~~~~~~~~~~~~~~~~~~~~~~

LOAD VERB LDV Debugger Verb Debugging Operation
Loads the verb that you wish to run and debug.
It simply prepares the verb to be run and debugged and then returns to the debug prompt.

LOAD VERB called VERBNAME

VERBNAME is the name of the verb that has to be loaded.

SEE ALSO: RUN VERB/DEBUG/
~~~~~~~~~~~~~~~~~~~~~~~~

RUN VERB RUN Debugger Verb Debugging Operation
Begins the execution of the verb loaded with LOAD VERB.
You may interrupt the running of the verb at any time by pressing <CONTROL-C>
twice. This will return you to the DEBUG prompt. If you use RUN VERB again
you will pick up where you left off. If you want to run the verb from the
beginning, you must LOAD VERB again. If you wish to see the program screen,
you may use VIEW PROGRAM SCREEN.
RUN VERB

SEE ALSO: LOAD VERB/VIEW PROGRAM SCREEN/DEBUG
~~~~~~~~~~~~~~~~~~~~~~~~

VIEW PROGRAM SCREEN VPS Debugger Verb Debugging Operation
Lets you see the program screen without having to resume the program with
RUN VERB.

VIEW PROGRAM SCREEN

SEE ALSO: RUN VERB/DEBUG/
~~~~~~~~~~~~~~~~~~~~~~~~

SET BREAKPOINT BP Debugger Verb Debugging Operation
Lets you set a breakpoint at the beginning of a verb.
You cannot set a breakpoint until you have first loaded a verb with LOAD VERB.
When executing, if you come upon a verb where there is a breakpoint set, exe-
cution is suspended and you return to the DEBUG prompt. You may examine
nouns, etc., and the resume the execution with RUN VERB. The breakpoints stay
put until removed. To remove a breakpoint, use REMOVE BREAKPOINT.
SET BREAKPOINT at verb VERBNAME
~~~~~~~~~~~~~~~~~~~~~~~~

VERBNAME is the name of the verb where you want to set the breakpoint.

SEE ALSO: REMOVE BEAKPOINT/RUN VERB/DEBUG
~~~~~~~~~~~~~~~~~~~~~~~~~

REMOVE BREAKPOINT RBP Debugger Verb Debugging Operation
Removes a breakpoint that was set with SET BREAKPOINT.

REMOVE BREAKPOINT at verb VERBNAME

VERBNAME is the name of the verb where you want to remove the breakpoint.

SEE ALSO: SET BREAKPOINT/RUN VERB/DEBUG
~~~~~~~~~~~~~~~~~~~~~~~~~

FILE STATUS FST Standard Noun Noun
Used by the verbs that perform file operations (external and regular).
It contains a numeric status of the most recent file operation. This status
tells you whether or not the file operation completed properly.
Both FILE STATUS and EXTERNAL ECHO are set after file operations, and either
may be checked. FILE STATUS, since it contains numeric values, is language
independent.
DISPLAY the value FILE STATUS

SEE ALSO: EXTERNAL ECHO/IPC STATUS/
The noun FILE STATUS (FST) contains a numeric status after all file operations (except those using mock internal emulation; mockflag = 1).

Both FILE STATUS and EXTERNAL ECHO are set after file operations, and either may be checked. FILE STATUS, since it contains numeric values, is language independent; whereas EXTERNAL ECHO contains text which is different for each country. If EXTERNAL ECHO is used, the verb is dependent upon the file status messages selected in PRAGMA.DES.

These are the possible values of FILE STATUS :

OK	0
record not found	1
duplicate reference	2

conflict	3
file is locked	4
queue is empty	5
queue is locked	6
prepare to stop	7
NO SUCH FILE	8
REFERENCE TOO LONG	9
MORE NOUNS IN LOCAL FILE	10
FEWER NOUNS IN LOCAL FILE	11
NO SUCH QUEUE	12
NO RESPONSE RECEIVED	13
EXTERNAL DISC IS FULL	14
TRANSMISSION ERROR	15
INTERNAL LOGIC ERROR	16
UNKNOWN EXTERNAL ECHO?	17

BTRIEVE

If you are using BTRIEVE as filemanager and the result of a fileoperation is INTERNAL LOGIC ERROR for the EXTERNAL ECHO, the FILE STATUS will contain the BTRIEVE error number. This is to help to trace BTRIEVE errors.

~~~~~~~~~~~~~~~~~~~~~~~~~

CLOSE ALL FILES CAF Standard Verb File Access Operation
Releases all records and closes all PFM and BTRIEVE files.
The PRVOCAB, PRMSG and PRSTD files are also closed.

Any ASCII file opened with a DOS OPEN verb is NOT closed. Neither are files
that have been opened with a SYSTEM CALL.

CLOSE ALL FILES

SEE ALSO: CLOSE FILE/EXTERNAL CLOSE FILE/SYSTEM CALL
CLOSE ALL FILES (CAF)
CLOSE FILE (CF)
EXTERNAL CLOSE FILE (XCF)

The verb CLOSE ALL FILES closes all open PFM files and Btrieve files and also releases all records of these files that may have been locked.

CLOSE FILE closes one specific file on the internal file path, whereas EXTERNAL CLOSE FILE closes
~~~~~~~~~~~~~~~~~~~~~~~~~

one specific file on the external file path. Any locked record of that file will be released.

Be aware that when closing a file the THIS pointer of that file is lost. For instance, you cannot get a record of a file, close that file and then save the modified record. PRAGMA will have lost the information of which record you wanted to update.

In addition to closing all the open user files, CLOSE ALL FILES also closes the files PRVOCAB.PFM, PRMSG.PFM and PRSTD.PFM, but reopens them immediately. But any ASCII file opened with a DOS OPEN verb (DOS OPEN APPEND, DOS OPEN CREATE and DOS OPEN READ) will not be closed. Nor will any file that might have been opened by a SYSTEM CALL be closed.

Usually in PRAGMA you don't have to be concerned with opening or closing files. The only exceptions are ASCII files opened with the DOS OPEN verbs.

These verbs for closing files are only to be used in special circumstances as when you want to ensure that an important file is shut as a precaution against file corruption in case of an inorderly shutdown of PRAGMA. Or if the operating system or Btrieve won't let you keep more than a certain amount of files open.

Another reason to close a file or all files is to force the operating system to write the data to disk. But be aware that even if you close files in PRAGMA, the operating system may not immediately perform the task if a cache is installed.

MULTIUSER ENVIRONMENT

But what happens in a multiuser environment like UNIX or in a network, where files are accessed by more than one user? Does closing a file within PRAGMA really close all the files?

UNIX has a data space for each user. In that data space UNIX records what files are open, what locks are pending, etc. All of this is done on a per user basis. So, just as UNLOCK ALL only unlocks what you have locked and nothing else, so closing a file only closes what you have opened.

If five people have the same file open, for instance, and one user does closes that file, that file only gets closed for that one user. As far as the other users are concerned, nothing has happened at all.

From the operating system point of view, the file is not really closed until all users that have it open have closed it. But to a user, it is not known if other users have a file open or not. He can only close the file for himself, and not for anyone else.

The effect of closing a file is only to write back to the operating system all of your changes. It does not totally close the file from the system point of view.

So, the important point in a multiuser system is that you need to have two points of view. That of the end user and that of the operating system.

The Novell network server handles files accessed by more than one user in a similar way to UNIX.

Usually this is not important. But if you want to write a backup routine and invoke it from within PRAGMA, be very careful. You should not do it for a multiuser environment. If you attempt to backup a file that is open for another user, it will not work, or at best, it will work very slowly and cause a slowdown of everything.

~~~~~~~~~~~~~~~~~~~~~~~~

SET FILE PATH FP Standard Verb File Access Operation
Sets the directory where all the files reside.
When doing a SET FILE PATH a CLOSE ALL FILES is first performed.
There can be only one active path open at a time per set of file words
(internal or external). The default is no path (current directory) or the
INTERNAL FILE PATH = set via PRAGMA.DES.

SET FILE PATH to the directory PATHNAME

PATHNAME is joined as is to the beginning of the file short name. Normally,
this path should end with a slash. Under DOS and OS/2 you may also include
a drive letter. When PATHNAME is a null string (""), the current directory
is selected.
SEE ALSO: SET EXTERNAL FILE PATH/CLOSE ALL FILES/
The standard verbs SET FILE PATH (FP), and SET EXTERNAL FILE PATH (XFP) allow you to set the directory where the files reside.

The first thing these two verbs do is a CLOSE ALL FILES. All records are released, etc.

Then the path specified is selected as the directory where files will be opened. There is a separate path for 'internal' file words and 'external' file words.

There can be only one active path open at a time per set of file words ('internal' or 'external'). Multiple paths at once are not possible, as the files are all closed when selecting a new path.

This path is joined as is to the beginning of the file short name. Normally, this path should end with a slash. Under DOS and OS/2, you may also include a drive letter. The default is no path (current directory). The current directory may also be selected with these verbs by giving a null string as the path (SET FILE PATH to the directory "").

In addition, a default path for 'internal' and 'external' files may be set via PRAGMA.DES. Two new strings are used to do this; 'INTERNAL FILE PATH = ' and 'EXTERNAL FILE PATH = '.

~~~~~~~~~~~~~~~~~~~~~~~~

SET EXTERNAL FILE PAXFP Standard Verb File Access Operation
Sets the directory where all the files reside.
When doing a SET EXTERNAL FILE PATH a CLOSE ALL FILES is first performed.

There can be only one active path open at a time per set of file words
(internal or external). The default is no path (current directory) or the
EXTERNAL FILE PATH = set via PRAGMA.DES.

SET EXTERNAL FILE PATH to the directory PATHNAME

PATHNAME is joined as is to the beginning of the file short name. Normally,
this path should end with a slash. Under DOS and OS/2 you may also include
a drive letter. When PATHNAME is a null string (""), the current directory
is selected.
SEE ALSO: SET FILE PATH/CLOSE ALL FILES/
The standard verbs SET FILE PATH (FP), and SET EXTERNAL FILE PATH (XFP) allow you to set the
directory where the files reside.

The first thing these two verbs do is a CLOSE ALL FILES. All records are released, etc.

Then the path specified is selected as the directory where files will be opened. There is a
separate path for 'internal' file words and 'external' file words.

There can be only one active path open at a time per set of file words ('internal' or 'external').
Multiple paths at once are not possible, as the files are all closed when selecting a new path.

This path is joined as is to the beginning of the file short name. Normally, this path should end
with a slash. Under DOS and OS/2, you may also include a drive letter. The default is no path
(current directory). The current directory may also be selected with these verbs by giving a null
string as the path (SET FILE PATH to the directory "").

In addition, a default path for 'internal' and 'external' files may be set via PRAGMA.DES. Two new
strings are used to do this; 'INTERNAL FILE PATH = ' and 'EXTERNAL FILE PATH = '.

~~~~~~~~~~~~~~~~~~~~~~~~~

GET WAIT LOCK FIRST GWLF Standard Verb File Access Operation
Retrieves the first record from a file and locks the record. If the record is
already locked by another user, it waits until the record becomes free.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

GET WAIT LOCK FIRST in file FILE

FILE is the name of the file whose FIRST record is to be accessed and locked.

SEE ALSO: GET NOWAIT LOCK FIRST/GET WAIT FIRST/GET NOWAIT FIRST
~~~~~~~~~~~~~~~~~~~~~~~~~

GET WAIT LOCK NEXT GWLN Standard Verb File Access Operation
Retrieves the next record from a file and locks the record. If the record is

already locked by another user, it waits until the record becomes free.
This is particularly useful after having positioned the file pointer directly
ahead of a record with the verb BEGIN.
The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.
GET WAIT LOCK NEXT in file FILE

FILE is the name of the file whose NEXT record is to be accessed and locked.

SEE ALSO: GET NOWAIT LOCK NEXT/GET WAIT FIRST/BEGIN
~~~~~~~~~~~~~~~~~~~~~~~~~~

GET WAIT LOCK THIS GWLT Standard Verb File Access Operation
Retrieves the present record from a file and locks the record. If the record
is already locked by another user, it waits until the record becomes free.
This is particularly useful after having positioned the file pointer directly
ahead of a record with the verb BEGIN THIS, enabling you to retieve the
previous record in a file.
The reference of the record is written to the target noun REFERENCE.
GET WAIT LOCK THIS in file FILE

FILE is the name of the file whose current record is to be accessed and
locked.

SEE ALSO: GET NOWAIT LOCK THIS/GET WAIT/BEGIN THIS
~~~~~~~~~~~~~~~~~~~~~~~~~~

GET WAIT LOCK LAST GWLL Standard Verb File Access Operation
Retrieves the last record from a file and locks the record. If the record is
already locked by another user, it waits until the record becomes free.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

GET WAIT LOCK LAST in file FILE

FILE is the name of the file whose LAST record is to be accessed and locked.

SEE ALSO: GET NOWAIT LOCK LAST/GET WAIT LAST/GET NOWAIT LAST
~~~~~~~~~~~~~~~~~~~~~~~~~~

GET WAIT LOCK   GWLK Standard Verb File Access Operation
Retrieves a record from a file and locks the record. If the record is already
locked by another user, it waits until the record becomes free.
If PRAGMA cannot find the specified record, the EXTERNAL ECHO will contain a
"record not found" message and no record will be retrieved.
After accessing the file the reference of the record is written to the target
~~~~~~~~~~~~~~~~~~~~~~~~~~

noun REFERENCE. EXTERNAL ECHO contains the status message of the access.
GET WAIT LOCK reference CHOICE in file FILE

CHOICE can be a noun, number or an expression that identifies the record you want to access. FILE is the name of the file that is to be accessed.

SEE ALSO: GET NOWAIT LOCK/GET WAIT LOCK THIS/BEGIN
~~~~~~~~~~~~~~~~~~~~~~~~~

GET NOWAIT LOCK FIRSGNLF Standard Verb File Access Operation
Retrieves the first record from a file and locks the record. If the record is already locked by another user, it immediately returns with a message of "conflict" in EXTERNAL ECHO and a status of 3 in FILE STATUS.

The reference of the record is written to the target noun REFERENCE.

GET NOWAIT LOCK FIRST in file FILE

FILE is the name of the file whose FIRST record is to be accessed and locked.

SEE ALSO: GET WAIT LOCK FIRST/GET NOWAIT FIRST/GET WAIT FIRST
~~~~~~~~~~~~~~~~~~~~~~~~~

GET NOWAIT LOCK NEXTGNLN Standard Verb File Access Operation
Retrieves the next record from a file and locks the record. If the record is already locked by another user, it immediately returns with a message of "conflict" in EXTERNAL ECHO and a status of 3 in FILE STATUS.
This is particularly useful after having positioned the file pointer directly ahead of a record with the verb BEGIN.
The reference of the record is written to the target noun REFERENCE.
GET NOWAIT LOCK NEXT in file FILE

FILE is the name of the file whose NEXT record is to be accessed and locked.

SEE ALSO: GET WAIT LOCK NEXT/GET NOWAIT FIRST/BEGIN
~~~~~~~~~~~~~~~~~~~~~~~~~

GET NOWAIT LOCK THISGNLT Standard Verb File Access Operation
Retrieves the present record from a file and locks the record. If the record is already locked by another user, it immediately returns with a message of "conflict" in EXTERNAL ECHO and a status of 3 in FILE STATUS.
This is particularly useful after having positioned the file pointer directly ahead of a record with the verb BEGIN THIS, enabling you to retrieve the previous record in a file.
GET NOWAIT LOCK THIS in file FILE

FILE is the name of the file whose current record is to be accessed and
~~~~~~~~~~~~~~~~~~~~~~~~~

locked.

SEE ALSO: GET WAIT LOCK THIS/GET NOWAIT/BEGIN THIS
~~~~~~~~~~~~~~~~~~~~~~~~~~

GET NOWAIT LOCK LASTGNLL Standard Verb File Access Operation
Retrieves the last record from a file and locks the record. If the record is already locked by another user, it immediately returns with a message of "conflict" in EXTERNAL ECHO and a status of 3 in FILE STATUS.
This is particularly useful after having positioned the file pointer directly ahead of a record with the verb BEGIN THIS, enabling you to retrieve the previous record in a file.
GET NOWAIT LOCK LAST in file FILE

FILE is the name of the file whose last record is to be accessed and locked.

SEE ALSO: GET WAIT LOCK THIS/GET NOWAIT/BEGIN THIS
~~~~~~~~~~~~~~~~~~~~~~~~~~

GET NOWAIT LOCK   GNLK Standard Verb File Access Operation
Retrieves a record from a file and locks the record. If the record is already locked by another user, it immediately returns with a message of "conflict" in EXTERNAL ECHO and a status of 3 in FILE STATUS.
If PRAGMA cannot find the specified record, the EXTERNAL ECHO will contain a "record not found" message and no record will be retrieved. After accessing the file the reference of the record is written to the target noun REFERENCE.
GET NOWAIT LOCK reference CHOICE in file FILE

CHOICE can be a noun, number or an expression that identifies the record you want to access. FILE is the name of the file that is to be accessed.

SEE ALSO: GET WAIT LOCK/GET NOWAIT LOCK THIS/BEGIN
~~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET WAIT LOXGWLFStandard Verb External File Access Operation
Retrieves the first record from a file and locks the record. If the record is already locked by another user, it waits until the record becomes free.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

EXTERNAL GET WAIT LOCK FIRST in file FILE

FILE is the name of the file whose FIRST record is to be accessed and locked.

SEE ALSO: EXTERNAL GET NOWAIT LOCK FIRST/EXTERNAL GET WAIT FIRST/EXTERNAL GET NOWAIT FIRST
~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET WAIT LOXGWLNStandard Verb External File Access Operation
Retrieves the next record from a file and locks the record. If the record is
already locked by another user, it waits until the record becomes free.
This is particularly useful after having positioned the file pointer directly
ahead of a record with the verb EXTERNAL BEGIN.
The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.
EXTERNAL GET WAIT LOCK NEXT in file FILE

FILE is the name of the file whose NEXT record is to be accessed and locked.

SEE ALSO: EXTERNAL GET NOWAIT LOCK NEXT/EXTERNAL GET WAIT FIRST/EXTERNAL BEGIN
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET WAIT LOXGWLTStandard Verb External File Access Operation
Retrieves the present record from a file and locks the record. If the record
is already locked by another user, it waits until the record becomes free.
This is particularly useful after having positioned the file pointer directly
ahead of a record with the verb EXTERNAL BEGIN THIS, enabling you to retieve
the previous record in a file.
The reference of the record is written to the target noun REFERENCE.
EXTERNAL GET WAIT LOCK THIS in file FILE

FILE is the name of the file whose current record is to be accessed and
locked.

SEE ALSO: EXTERNAL GET NOWAIT LOCK THIS/EXTERNAL GET WAIT/EXTERNAL BEGIN THIS
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET WAIT LOXGWLLStandard Verb External File Access Operation
Retrieves the last record from a file and locks the record. If the record is
already locked by another user, it waits until the record becomes free.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

EXTERNAL GET WAIT LOCK LAST in file FILE

FILE is the name of the file whose LAST record is to be accessed and locked.

SEE ALSO: EXTERNAL GET NOWAIT LOCK LAST/EXTERNAL GET WAIT LAST/EXTERNAL GET NOWAIT LAST
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET WAIT LOXGWLKStandard Verb External File Access Operation
Retrieves a record from a file and locks the record. If the record is already

locked by another user, it waits until the record becomes free.
If PRAGMA cannot find the specified record, the EXTERNAL ECHO will contain a
"record not found" message and no record will be retrieved.
After accessing the file the reference of the record is written to the target
noun REFERENCE. EXTERNAL ECHO contains the status message of the access.
EXTERNAL GET WAIT LOCK reference CHOICE in file FILE

CHOICE can be a noun, number or an expression that identifies the record
you want to access. FILE is the name of the file that is to be accessed.

SEE ALSO: EXTERNAL GET NOWAIT LOCK/EXTERNAL GET WAIT LOCK THIS/EXTERNAL BEGIN
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET NOWAIT XGNLFStandard Verb External File Access Operation
Retrieves the first record from a file and locks the record. If the record is
already locked by another user, it immediately returns with a message of
"conflict" in EXTERNAL ECHO and a status of 3 in FILE STATUS.

The reference of the record is written to the target noun REFERENCE.

EXTERNAL GET NOWAIT LOCK FIRST in file FILE

FILE is the name of the file whose FIRST record is to be accessed and locked.

SEE ALSO: EXTERNAL GET WAIT LOCK FIRST/EXTERNAL GET NOWAIT FIRST/EXTERNAL GET WAIT FIRST
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET NOWAIT XGNLNStandard Verb External File Access Operation
Retrieves the next record from a file and locks the record. If the record is
already locked by another user, it immediately returns with a message of
"conflict" in EXTERNAL ECHO and a status of 3 in FILE STATUS.
This is particularly useful after having positioned the file pointer directly
ahead of a record with the verb EXTERNAL BEGIN.
The reference of the record is written to the target noun REFERENCE.
EXTERNAL GET NOWAIT LOCK NEXT in file FILE

FILE is the name of the file whose NEXT record is to be accessed and locked.

SEE ALSO: EXTERNAL GET WAIT LOCK NEXT/EXTERNAL GET NOWAIT FIRST/EXTERNAL BEGIN
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET NOWAIT XGNLTStandard Verb External File Access Operation
Retrieves the present record from a file and locks the record. If the record
is already locked by another user, it immediately returns with a message of
"conflict" in EXTERNAL ECHO and a status of 3 in FILE STATUS.
This is particularly useful after having positioned the file pointer directly
ahead of a record with the verb EXTERNAL BEGIN THIS, enabling you to retrieve
~~~~~~~~~~~~~~~~~~~~~~~~~

the previous record in a file.
EXTERNAL GET NOWAIT LOCK THIS in file FILE

FILE is the name of the file whose current record is to be accessed and locked.

SEE ALSO: EXTERNAL GET WAIT LOCK THIS/EXTERNAL GET NOWAIT/EXTERNAL BEGIN THIS
~~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET NOWAIT XGNLLStandard Verb External File Access Operation
Retrieves the last record from a file and locks the record. If the record is already locked by another user, it immediately returns with a message of "conflict" in EXTERNAL ECHO and a status of 3 in FILE STATUS.

EXTERNAL GET NOWAIT LOCK LAST in file FILE

FILE is the name of the file whose last record is to be accessed and locked.

SEE ALSO: EXTERNAL GET WAIT LOCK THIS/EXTERNAL GET NOWAIT/EXTERNAL BEGIN THIS
~~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET NOWAIT XGNLKStandard Verb External File Access Operation
Retrieves a record from a file and locks the record. If the record is already locked by another user, it immediately returns with a message of "conflict" in EXTERNAL ECHO and a status of 3 in FILE STATUS.
If PRAGMA cannot find the specified record, the EXTERNAL ECHO will contain a "record not found" message and no record will be retrieved. After accessing the file the reference of the record is written to the target noun REFERENCE.
EXTERNAL GET NOWAIT LOCK reference CHOICE in file FILE

CHOICE can be a noun, number or an expression that identifies the record you want to access. FILE is the name of the file that is to be accessed.

SEE ALSO: EXTERNAL GET WAIT LOCK/EXTERNAL GET NOWAIT LOCK THIS/EXTERNAL BEGIN
~~~~~~~~~~~~~~~~~~~~~~~~~~

UNLOCK RECORD   UR   Standard Verb File Access Operation
Unlocks a specified single record in a specified file.
It is only possible to unlock a record locked by the same user. You cannot unlock records locked by another user.
A successful unlock will return an "OK" status to EXTERNAL ECHO (FST = 0).
If you try to unlock a record that is not locked, you will get a "record not found" in EXTERNAL ECHO (FST = 1).
UNLOCK RECORD reference CHOICE in file FILE

CHOICE can ba a noun, number or an expression that identifies the record you
~~~~~~~~~~~~~~~~~~~~~~~~~~

want to unlock. FILE is the name of the file that is to be accessed.

SEE ALSO: UNLOCK FILE/UNLOCK ALL/EXTERNAL UNLOCK RECORD
did not get praref2 186
The standard verbs SET FILE PATH (FP), and SET EXTERNAL FILE PATH (XFP) allow you to set the directory where the files reside.

The first thing these two verbs do is a CLOSE ALL FILES. All records are released, etc.

Then the path specified is selected as the directory where files will be opened. There is a separate path for 'internal' file words and 'external' file words.

There can be only one active path open at a time per set of file words ('internal' or 'external'). Multiple paths at once are not possible, as the files are all closed when selecting a new path.

This path is joined as is to the beginning of the file short name. Normally, this path should end with a slash. Under DOS and OS/2, you may also include a drive letter. The default is no path (current directory). The current directory may also be selected with these verbs by giving a null string as the path (SET FILE PATH to the directory "").

In addition, a default path for 'internal' and 'external' files may be set via PRAGMA.DES. Two new strings are used to do this; 'INTERNAL FILE PATH = ' and 'EXTERNAL FILE PATH = '.

~~~~~~~~~~~~~~~~~~~~~~~~~

UNLOCK FILE   UF   Standard Verb File Access Operation
Unlocks all locked records in a single file.
It is only possible to unlock records in a file locked by the same user.
You cannot unlock records locked by another user.
A successful unlock will return an "OK" status to EXTERNAL ECHO (FST = 0).
If you try to unlock a file that has no locked records, you will get a "record not found" in EXTERNAL ECHO (FST = 1).
UNLOCK FILE called FILE

FILE is the name of the file whose records are to be unlocked.

SEE ALSO: UNLOCK RECORD/UNLOCK ALL/EXTERNAL UNLOCK FILE
~~~~~~~~~~~~~~~~~~~~~~~~~

UNLOCK ALL   UA   Standard Verb File Access Operation
Unlocks all locked records in all files, but only files of the respective file manager. It is only possible to unlock records in a file locked by the same user. You cannot unlock records locked by another user.
A successful unlock will return an "OK" status to EXTERNAL ECHO (FST = 0).
If you try do an UNLOCK ALL and there are no locked records, you will get a "record not found" in EXTERNAL ECHO (FST = 1).
UNLOCK ALL

SEE ALSO: UNLOCK RECORD/UNLOCK FILE/EXTERNAL UNLOCK ALL
~~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL UNLOCK RECOXUR Standard Verb External File Access Operation
Unlocks a specified single record in a specified file.
It is only possible to unlock a record locked by the same user. You cannot unlock records locked by another user.
A successful unlock will return an "OK" status to EXTERNAL ECHO (FST = 0).
If you try to unlock a record that is not locked, you will get a "record not found" in EXTERNAL ECHO (FST = 1).
EXTERNAL UNLOCK RECORD reference CHOICE in file FILE

CHOICE can be a noun, number or an expression that identifies the record you want to unlock. FILE is the name of the file that is to be accessed.

SEE ALSO: EXTERNAL UNLOCK FILE/EXTERNAL UNLOCK ALL/UNLOCK RECORD
~~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL UNLOCK FILEXUF Standard Verb External File Access Operation
Unlocks all locked records in a single file.
It is only possible to unlock records in a file locked by the same user.
You cannot unlock records locked by another user.
A successful unlock will return an "OK" status to EXTERNAL ECHO (FST = 0).
If you try to unlock a file that has no locked records, you will get a "record not found" in EXTERNAL ECHO (FST = 1).
EXTERNAL UNLOCK FILE called FILE

FILE is the name of the file whose records are to be unlocked.

SEE ALSO: EXTERNAL UNLOCK RECORD/EXTERNAL UNLOCK ALL/UNLOCK FILE
~~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL UNLOCK ALL XUA Standard Verb External File Access Operation
Unlocks all locked records in all files, but only files of the respective file manager. It is only possible to unlock records in a file locked by the same user. You cannot unlock records locked by another user.
A successful unlock will return an "OK" status to EXTERNAL ECHO (FST = 0).
If you try do an UNLOCK ALL and there are no locked records, you will get a "record not found" in EXTERNAL ECHO (FST = 1).
EXTERNAL UNLOCK ALL
~~~~~~~~~~~~~~~~~~~~~~~~~~

SEE ALSO: EXTERNAL UNLOCK RECORD/EXTERNAL UNLOCK FILE/UNLOCK ALL
~~~~~~~~~~~~~~~~~~~~~~~~~

GET WAIT FIRST GWF Standard Verb File Access Operation
Retrieves the first record from a file. If the record is locked by another user, it waits until the record becomes free.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

GET WAIT FIRST in file FILE

FILE is the name of the file whose FIRST record is to be accessed.

SEE ALSO: GET NOWAIT FIRST/GET WAIT LOCK FIRST/GET NOWAIT LOCK FIRST
~~~~~~~~~~~~~~~~~~~~~~~~~

GET WAIT NEXT GWN Standard Verb File Access Operation
Retrieves the next record from a file. If the record is locked by another user, it waits until the record becomes free.
This is particularly useful after having positioned the file pointer directly ahead of a record with the verb BEGIN.
The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.
GET WAIT NEXT in file FILE

FILE is the name of the file whose NEXT record is to be accessed.

SEE ALSO: GET NOWAIT NEXT/GET WAIT LOCK NEXT/BEGIN
~~~~~~~~~~~~~~~~~~~~~~~~~

GET WAIT THIS GWT Standard Verb File Access Operation
Retrieves the present record from a file. If the record is locked by another user, it waits until the record becomes free.
This is particularly useful after having positioned the file pointer directly ahead of a record with the verb BEGIN THIS, enabling you to retieve the previous record in a file.
The reference of the record is written to the target noun REFERENCE.
GET WAIT THIS in file FILE

FILE is the name of the file to be accessed.

SEE ALSO: GET NOWAIT THIS/GET WAIT LOCK/BEGIN THIS
~~~~~~~~~~~~~~~~~~~~~~~~~

GET WAIT LAST   GWL Standard Verb File Access Operation
Retrieves the last record from a file. If the record is already locked by
by another user, it waits until the record becomes free.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

GET WAIT LAST in file FILE

FILE is the name of the file whose LAST record is to be accessed.

SEE ALSO: GET NOWAIT LAST/GET WAIT LOCK LAST/GET NOWAIT LOCK LAST
~~~~~~~~~~~~~~~~~~~~~~~~~

GET WAIT   GW   Standard Verb File Access Operation
Retrieves a record from a file. If the record is locked by another user,
it waits until the record becomes free.
If PRAGMA cannot find the specified record, the EXTERNAL ECHO will contain a
"record not found" message and no record will be retrieved.
After accessing the file the reference of the record is written to the target
noun REFERENCE. EXTERNAL ECHO contains the status message of the access.
GET WAIT reference CHOICE in file FILE

CHOICE can be a noun, number or an expression that identifies the record
you want to access. FILE is the name of the file that is to be accessed.

SEE ALSO: GET NOWAIT/GET WAIT THIS/BEGIN
~~~~~~~~~~~~~~~~~~~~~~~~~

GET NOWAIT FIRST   GNF Standard Verb File Access Operation
Retrieves the first record from a file. If the record is locked by another
user, it immediately returns with a message of "conflict" in EXTERNAL ECHO
and a status of 3 in FILE STATUS.

The reference of the record is written to the target noun REFERENCE.

GET NOWAIT FIRST in file FILE

FILE is the name of the file whose FIRST record is to be accessed.

SEE ALSO: GET WAIT FIRST/GET NOWAIT LOCK FIRST/GET WAIT LOCK FIRST
~~~~~~~~~~~~~~~~~~~~~~~~~

GET NOWAIT NEXT   GNN Standard Verb File Access Operation
Retrieves the next record from a file. If the record is locked by another
user, it immediately returns with a message of "conflict" in EXTERNAL ECHO
and a status of 3 in FILE STATUS.
~~~~~~~~~~~~~~~~~~~~~~~~~

This is particularly useful after having positioned the file pointer directly ahead of a record with the verb BEGIN.
The reference of the record is written to the target noun REFERENCE.
GET NOWAIT NEXT in file FILE

FILE is the name of the file whose NEXT record is to be accessed.

SEE ALSO: GET WAIT NEXT/GET NOWAIT LOCK NEXT/BEGIN
~~~~~~~~~~~~~~~~~~~~~~~~~

GET NOWAIT THIS GNT Standard Verb File Access Operation
Retrieves the present record from a file. If the record is locked by another user, it immediately returns with a message of "conflict" in EXTERNAL ECHO and a status of 3 in FILE STATUS.
This is particularly useful after having positioned the file pointer directly ahead of a record with the verb BEGIN THIS, enabling you to retrieve the previous record in a file.
GET NOWAIT THIS in file FILE

FILE is the name of the file to be accessed.

SEE ALSO: GET WAIT THIS/GET NOWAIT LOCK THIS/BEGIN THIS
~~~~~~~~~~~~~~~~~~~~~~~~~

GET NOWAIT LAST GNL Standard Verb File Access Operation
Retrieves the last record from a file. If the record is locked by another user, it immediately returns with a message of "conflict" in EXTERNAL ECHO and a status of 3 in FILE STATUS.

GET NOWAIT LAST in file FILE

FILE is the name of the file to be accessed.

SEE ALSO: GET WAIT LAST/GET NOWAIT LOCK LAST/BEGIN THIS
~~~~~~~~~~~~~~~~~~~~~~~~~

GET NOWAIT GNW Standard Verb File Access Operation
Retrieves a record from a file. If the record is locked by another user, it immediately returns with a message of "conflict" in EXTERNAL ECHO and a status of 3 in FILE STATUS.
If PRAGMA cannot find the specified record, the EXTERNAL ECHO will contain a "record not found" message and no record will be retrieved. After accessing the file the reference of the record is written to the target noun REFERENCE.
GET NOWAIT reference CHOICE in file FILE
~~~~~~~~~~~~~~~~~~~~~~~~~

CHOICE can be a noun, number or an expression that identifies the record you
want to access. FILE is the name of the file that is to be accessed.

SEE ALSO: GET WAIT/GET NOWAIT LOCK/BEGIN
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET WAIT FIXGWF Standard Verb External File Access Operation
Retrieves the first record from a file. If the record is locked by another
user, it waits until the record becomes free.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

EXTERNAL GET WAIT FIRST in file FILE

FILE is the name of the file whose FIRST record is to be accessed.

SEE ALSO: EXTERNAL GET NOWAIT FIRST/EXTERNAL GET WAIT LOCK FIRST/EXTERNAL GET NOWAIT LOCK FIRST
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET WAIT NEXGWN Standard Verb External File Access Operation
Retrieves the next record from a file. If the record is locked by another
user, it waits until the record becomes free.
This is particularly useful after having positioned the file pointer directly
ahead of a record with the verb EXTERNAL BEGIN.
The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.
EXTERNAL GET WAIT NEXT in file FILE

FILE is the name of the file whose NEXT record is to be accessed.

SEE ALSO: EXTERNAL GET NOWAIT NEXT/EXTERNAL GET WAIT LOCK NEXT/EXTERNAL BEGIN
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET WAIT THXGWT Standard Verb External File Access Operation
Retrieves the present record from a file. If the record is locked by another
user, it waits until the record becomes free.
This is particularly useful after having positioned the file pointer directly
ahead of a record with the verb EXTERNAL BEGIN THIS, enabling you to retieve
previous record in a file.
The reference of the record is written to the target noun REFERENCE.
EXTERNAL GET WAIT THIS in file FILE

FILE is the name of the file to be accessed.
~~~~~~~~~~~~~~~~~~~~~~~~~

SEE ALSO: EXTERNAL GET NOWAIT THIS/EXTERNAL GET WAIT LOCK/EXTERNAL BEGIN THIS
~~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET WAIT LAXGWL Standard Verb External File Access Operation
Retrieves the last record from a file. If the record is already locked by
by another user, it waits until the record becomes free.

The reference of the record is written to the target noun REFERENCE.
The noun EXTERNAL ECHO contains the status message of the file operation.

EXTERNAL GET WAIT LAST in file FILE

FILE is the name of the file whose LAST record is to be accessed.

SEE ALSO: EXTERNAL GET NOWAIT LAST/EXTERNAL GET WAIT LOCK LAST/EXTERNAL GET NOWAIT LOCK LAST
~~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET WAIT   XGW Standard Verb External File Access Operation
Retrieves a record from a file. If the record is locked by another user,
it waits until the record becomes free.
If PRAGMA cannot find the specified record, the EXTERNAL ECHO will contain a
"record not found" message and no record will be retrieved.
After accessing the file the reference of the record is written to the target
noun REFERENCE. EXTERNAL ECHO contains the status message of the access.
EXTERNAL GET WAIT reference CHOICE in file FILE

CHOICE can be a noun, number or an expression that identifies the record
you want to access. FILE is the name of the file that is to be accessed.

SEE ALSO: EXTERNAL GET NOWAIT/EXTERNAL GET WAIT THIS/EXTERNAL BEGIN
~~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET NOWAIT XGNF Standard Verb External File Access Operation
Retrieves the first record from a file. If the record is locked by another
user, it immediately returns with a message of "conflict" in EXTERNAL ECHO
and a status of 3 in FILE STATUS.

The reference of the record is written to the target noun REFERENCE.

EXTERNAL GET NOWAIT FIRST in file FILE

FILE is the name of the file whose FIRST record is to be accessed.

SEE ALSO: EXTERNAL GET WAIT FIRST/EXTERNAL GET NOWAIT LOCK FIRST/EXTERNAL GET WAIT LOCK FIRST
~~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET NOWAIT XGNN Standard Verb External File Access Operation
Retrieves the next record from a file. If the record is locked by another user, it immediately returns with a message of "conflict" in EXTERNAL ECHO and a status of 3 in FILE STATUS.
This is particularly useful after having positioned the file pointer directly ahead of a record with the verb EXTERNAL BEGIN.
The reference of the record is written to the target noun REFERENCE.
EXTERNAL GET NOWAIT NEXT in file FILE

FILE is the name of the file whose NEXT record is to be accessed.

SEE ALSO: EXTERNAL GET WAIT NEXT/EXTERNAL GET NOWAIT LOCK NEXT/EXTERNAL BEGIN
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET NOWAIT XGNT Standard Verb External File Access Operation
Retrieves the present record from a file. If the record is locked by another user, it immediately returns with a message of "conflict" in EXTERNAL ECHO and a status of 3 in FILE STATUS.
This is particularly useful after having positioned the file pointer directly ahead of a record with the verb EXTERNAL BEGIN THIS, enabling you to retrieve the previous record in a file.
EXTERNAL GET NOWAIT THIS in file FILE

FILE is the name of the file to be accessed.

SEE ALSO: EXTERNAL GET WAIT THIS/EXTERNAL GET NOWAIT LOCK THIS/EXTERNAL BEGIN THIS
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET NOWAIT XGNL Standard Verb External File Access Operation
Retrieves the last record from a file. If the record is locked by another user, it immediately returns with a message of "conflict" in EXTERNAL ECHO and a status of 3 in FILE STATUS.

EXTERNAL GET NOWAIT LAST in file FILE

FILE is the name of the file to be accessed.

SEE ALSO: EXTERNAL GET WAIT LAST/EXTERNAL GET NOWAIT LOCK LAST/EXTERNAL BEGIN THIS
~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL GET NOWAIT XGNW Standard Verb External File Access Operation
Retrieves a record from a file. If the record is locked by another user, it immediately returns with a message of "conflict" in EXTERNAL ECHO and a status of 3 in FILE STATUS.
~~~~~~~~~~~~~~~~~~~~~~~~~

If PRAGMA cannot find the specified record, the EXTERNAL ECHO will contain a "record not found" message and no record will be retrieved. After accessing the file the reference of the record is written to the target noun REFERENCE.
EXTERNAL GET NOWAIT reference CHOICE in file FILE

CHOICE can be a noun, number or an expression that identifies the record you want to access. FILE is the name of the file that is to be accessed.

SEE ALSO: EXTERNAL GET WAIT/EXTERNAL GET NOWAIT LOCK/EXTERNAL BEGIN
~~~~~~~~~~~~~~~~~~~~~~~~~

READ SECURITY VALUE RSV Standard Verb Operating System Operation
Allows you to read certain areas of the security device. You cannot write to it under any circumstances.
Location 0 is the serial number of the device, locations 1 through 8 are user definable numbers which must be specified when ordering the devices.
Locations 10 through 12 contain other device parameters.

READ SECURITY VALUE location LOCATION NUMBER into the noun NOUN

LOCATION NUMBER is the number of the block location to test. Valid range 0 to 8 and 10 to 12 only.
NOUN is the noun you want the security value to be copied to.

SEE ALSO: REMOVE NAMES/RETURN TO OPERATING SYSTEM/
READ SECURITY VALUE allows you to read certain areas of the security block. You cannot write to the block under any circumstances.

The first parameter to READ SECURITY VALUE is the number of the block location to read. The valid range is from 0 through 8 and 10 through 12. An invalid location number will cause an invalid numeric value error.

Location zero is the serial number of the block. This is preset by Logical when a block is made, and each block is different. The serial number is also printed on the sticker attached to the block. You have no control over this number. But you can use it to serialize your applications, if that is desired. And you do not have to do anything special to use this number. Also note that a few early blocks did not have serial numbers, and this location contains a zero.

Locations 1 through 8 are user definable numbers. You can put anything you want into them, but you must tell us when ordering a block what you want there. Otherwise, all eight of these numbers will be zero. You cannot set these numbers yourself. Only Logical can do that when programming the block. We can reprogram them for you, but you have to send the block back to us for that.

The first of these eight locations is also what used to be the application number of the block. Old blocks that were ordered and had a special number in them for application security, had something in user location 1. If you attempt to read such a block, this is what you will see. Every other location (2-8) until now has contained zero.

Location 9 is reserved for future use. If you attempt to read this location it will cause an invalid numeric value error.
~~~~~~~~~~~~~~~~~~~~~~~~~

Location 10 is the security device type. The following is a partial list of the various types. Outdated types have been omitted.

2 PRAGMA 3 programmable
3 PRAGMA 3 runtime
4 Fileserver
9 UNIX runtime
10 UNIX programmable
13 PRAGMA 4 runtime
14 PRAGMA 4 programmable
25 UNIX compiled
26 PRAGMA 4 compiled

Location 11 is the country or dealer number. The following is a partial list of the various countries.

6 Italy
22 USA
26 Europe

Location 12 is the application number. This number is usually zero. It will contain a number if a user definable number in the locations 1 through 8 is being used.

Some serial Unix blocks for minicomputers (the green ones) have no memory in them, and thus have no serial number and no user definable locations. All of them will be zero if READ SECURITY VALUE is attempted.

CAREFUL:

READ SECURITY VALUE does not actually read the security device at the moment the verb is run. The values are read by PRAGMA at startup and passed along to READ SECURITY VALUE. That means, for example, that you cannot employ READ SECURITY VALUE to check various security devices, one after another.

WARNING:

TRACE will NOT show where READ SECURITY VALUE is used.

EXCHANGE and EXCHANGE ALL will NOT allow READ SECURITY VALUE to be substituted for something else.

TRANSLATE TO C will refuse to translate anything if it finds verbs containing calls to READ SECURITY VALUE.

RECAP and RECAP ALL will skip over any line where READ SECURITY VALUE is used. The only place where you can see a definition line containing READ SECURITY VALUE is inside the definition editor (this is necessary of course).

SELECTIVE SAVE is entirely unaffected by the security explained here. This may sound strange at first. But it is felt that if you truly put enough READ SECURITY VALUE calls about, that if it is possible to SELECTIVE SAVE them all away, then all of the important parts of your application must be gone too.

TIP:

In order to provide security for your application, it is necessary to put LOTS of calls to READ SECURITY VALUE all around your application. It is NOT necessary to call READ SECURITY VALUE every other thing you do, as that will just slow down processing to a crawl. Rather, put a counter noun in, and only call READ SECURITY VALUE every 10th or 20th time through a certain place. The important thing is to have a lot of these calls spread around your application like land mines. It is not necessary that the land mines go off the first time they are stepped on; it is only important that they are there. Sooner or later the traffic over them will cause one of them to activate if the security block is incorrect. That should be sufficient.

~~~~~~~~~~~~~~~~~~~~~~~~

REMOVE NAMES RMN Standard InteraGeneral Vocabulary Operation
Renames an application to have invisible names. The new names generated are
not only invisible, they are untypable and totally invalid in all places where
an item name is requested.
WARNING! Before using this utility MAKE A BACKUP of your vocab.
There is NO WAY that invisible names can be made visible again.
YOU HAVE BEEN WARNED!
REMOVE NAMES
Type top name of tree to rename
LV, LF, or LN to list items,
or RETURN if no more ->
etc.

SEE ALSO: CHECK SECURITY/SELECTIVE SAVE/
DISCUSSION

REMOVE NAMES renames an application to have invisible names.

The new names generated are not only invisible, they are untypable, and totally invalid in all places where an item name is requested. A verb thus processed can no longer be altered. Once items have had their names removed, they no longer show up on any listing operations, nor do they show up on a RECAP ALL. They seem to have vanished, yet they are still there. Also, when an item is processed, in addition to having its name removed, all label names, comments, object names, and object prompts are removed. None of this affects the operation of the verb. And of course, ALL of this activity is completely and totally IRREVERSIBLE.

REMOVE NAMES operates by processing trees of user items, similar to the way SELECTIVE SAVE and TRANSLATE TO C work. First, you "mark" trees of items that you want to have the names removed from. Generally, these will be parts of your application that you wish to keep prying eyes out of. For instance, those parts where you have put calls to CHECK SECURITY. Also, every item used by the items typed in (in other words, the full tree), gets marked to be processed too. You may mark as
~~~~~~~~~~~~~~~~~~~~~~~~

many trees this way as you wish. When you are done, a lone <RETURN> key will send you to the next step.

You are then asked if you want to UNMARK all of the files, and the nouns of their definitions. This may be useful if you only want to protect some verbs, but still want to allow others to write new verbs that access your files. It allows you to automatically unmark all files at once, without having to type in each file name one by one.

Next, you may UNMARK some trees to PREVENT them from having their names removed. This is useful in the case of utility verbs. You may only want to protect the application verbs, and leave the utility verbs used by them visible, so that others may use them in new verbs.

After unmarking as many trees as necessary, and then pressing a lone <RETURN>, you arrive at the final step. You may then UNMARK as many INDIVIDUAL items as required. These items (if they are verbs) are NOT processed as a tree, but ONLY the single items specified will by unmarked. This may also be used to unmark some files individually, in case you chose NOT to unmark all files automatically as described above. Files, when unmarked via this method ARE unmarked for their full tree (the tree of a file includes the nouns of its definition); thus you do not have to also unmark all of their nouns one by one. It may also be used to unmark individual verbs without also unmarking their full tree in case this is required.

When processing an application this way, it is necessary to remember to provide a valid verb name to be typed at START to allow one to enter the application. Remember that RMN will remove all names from the complete tree as marked. If you mark EVERYTHING this way, there will be NO valid verb names left that can be typed at START. So, for each application, or for each verb that is necessary to type at START, you should first make ANOTHER verb, just above it. These new verbs will have only one line; they will only call the real startup verb of the application (the old name you used to type at START). You then RENAME the old verb (the original startup verb) to be something else, and then RENAME the new verb to be the same as what the old name used to be. The new name (that now has the original old name) should then be preserved when performing a REMOVE NAMES. You will NOT type this name in when marking trees. Instead, the original verb (that now has some new name) is used to mark trees. Using this procedure, you will provide a startup verb name, that can be typed at START, and it will contain only one line, the name of the original verb (that used to be the startup verb, but was subsequently renamed), and that verb name will be blank. Thus, this startup verb cannot be altered or modified in some way to bypass any aspect of your security.

If you should try to TRANSLATE TO C some verbs that have had their names removed, the resulting C code will not compile. Of course, you will probably not even get to this step, as verbs without names probably have calls to CHECK SECURITY, and TRANSLATE TO C will not operate in this case. But if there were no CHS calls, this is what would happen.

WARNING Once you have performed a REMOVE NAMES you cannot show whatever you removed. Do NOT use REMOVE NAMES on your master application.

YOU HAVE BEEN WARNED.

~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL DELETE ALL XDA Standard Verb External File Access Operation
~~~~~~~~~~~~~~~~~~~~~~~~~

Deletes all the records in a file.
It does not affect the definition of the file.

The noun EXTERNAL ECHO contains the status message of the DELETE ALL opera-
tion.
Operates very fast. It is the best way to clear a file of records.
EXTERNAL DELETE ALL in file FILE

FILE is the name of the file whose records are to be deleted.

SEE ALSO: EXTERNAL DELETE/EXTERNAL DELETE THIS/DELETE ALL
DELETE ALL and EXTERNAL DELETE ALL delete all the records in a file, without affecting the definition of the file.

These two verbs operate very fast and this is the best way to clear a file of all its records.

The noun EXTERNAL ECHO or FILE STATUS contains the status message of the DELETE ALL or EXTERNAL DELETE ALL operations.

If any record has been locked by another user you will get back the status "conflict" in the NOUN EXTERNAL ECHO or a 3 in the noun FILE STATUS and the file will not be deleted.

If you are using the PRAGMA 3 internal emulation mode (mode 1) DELETE ALL and EXTERNAL DELETE ALL will wait until nobody has any pending locks and then proceed.

WARNING

DELETE_ALL and EXTERNAL DELETE ALL function by performing a create operation. This causes a problem if under DOS you use the DOS command APPEND and put your PFM or Btrieve files into a different directory than the generated .EXE file. When PFM or Btrieve look for a file, APPEND searches for the file and all is well. But when a DELETE_ALL performs a create operation, it automatically does it in the same directory where the program resides. And from then on all data is written and read from that newly created file.

BTRIEVE WARNING

When Btrieve attempts a DELETE ALL or an EXTERNAL DELETE ALL and encounters a locked record it may give an "INTERNAL LOGIC ERROR" instead of a "conflict". More testing is needed to clarify this situation.

~~~~~~~~~~~~~~~~~~~~~~~~~

CREATE PFM FILE   CPF Standard InteraDefinition Operation
Creates automatically an empty .PFM file on the internal file path.
~~~~~~~~~~~~~~~~~~~~~~~~~

This capability is only available if the file has a nonzero mockflag and only works for PFM. A .DES file is created only for completeness, since it is not really necessary. PRAGMA also warns you if one or both of these files already exist, and allows overwrite or abort.
CREATE PFM FILE for the file named FILE

FILE is the long PRAGMA name of the PFM file that is to be created.
The empty .PFM file that is created is called SHORTNAME.PFM.

SEE ALSO: EXTERNAL CREATE PFM FILE/FILE/
~~~~~~~~~~~~~~~~~~~~~~~~~

SAVE SCREEN WINDOW ISSWI Standard Verb Screen Operation
Saves into a noun a user defined window of a screen.
All the text, attributes and colors are saved.
The saved screen window can be placed anywhere on the screen since the image uses relative cursor positioning.
The image captured by SSWI is not compressed.

SAVE SCREEN WINDOW IMAGE starting row START ROW starting column START COLUMN ending row END ROW ending column END COLUMN to the noun NOUN

START ROW and START COLUMN represent the upper left corner of the window, END ROW and END COLUMN the right down corner of the window and may be nouns or constants. NOUN will contain all the information to display the window.
SEE ALSO: SAVE SCREEN IMAGE/SAVE SCREEN WINDOW TEXT/DISPLAY
~~~~~~~~~~~~~~~~~~~~~~~~~

SAVE SCREEN WINDOW TSSWT Standard Verb Screen Operation
Saves into a noun a user defined window of a screen.
Only the text is saved, the attributes and colors are not.
The saved screen window can be placed anywhere on the screen since the image uses relative cursor positioning.
The image captured by SSWT is not compressed.

SAVE SCREEN WINDOW IMAGE starting row START ROW starting column START COLUMN ending row END ROW ending column END COLUMN to the noun NOUN

START ROW and START COLUMN represent the upper left corner of the window, END ROW and END COLUMN the right down corner of the window and may be nouns or constants. NOUN will contain all the information to display the window.
SEE ALSO: SAVE SCREEN IMAGE/SAVE SCREEN WINDOW IMAGE/DISPLAY
~~~~~~~~~~~~~~~~~~~~~~~~~

DOS READ LAYOUT DRL Standard Verb Operating System Operation
After having opened with DOS OPEN READ the file that you want to read from, you can read a part of the contents of the file with the verb DOS READ LAYOUT.
DOS READ LAYOUT terminates reading when it has read the number of characters specified in the layout. The RECEIVE TERM CHAR will also terminate reading the file.
~~~~~~~~~~~~~~~~~~~~~~~~~

DOS OPEN READ filename FILENAME
DOS READ LAYOUT using layout NUMBER into the noun NOUN
DOS CLOSE READ
FILENAME is an expression or noun containing the name (and path) of the operating system file you want to read into the noun NOUN, with NUMBER the maximum number of characters that will be read before the read terminates.
SEE ALSO: DOS READ/INPUT LAYOUT/DOS OPEN READ
~~~~~~~~~~~~~~~~~~~~~~~~~

DOS WRITE LAYOUT DWL Standard Verb Operating System Operation
After having opened with DOS OPEN APPEND or DOS OPEN CREATE an operating system file, you can write formatted text and numbers to the file with the verb DOS WRITE. DOS WRITE LAYOUT operates like a combination of OUTPUT LAYOUT and DOS WRITE.

DOS OPEN APPEND filename FILENAME
DOS WRITE LAYOUT using layout NUMBER the value ITEM
DOS CLOSE WRITE
FILENAME is an expression or noun containing the name and optional path of the operating system file you want to write to. ITEM is the noun or expression you want to write, NUMBER is the maximum number of characters you can write.
SEE ALSO: DOS WRITE/OUTPUT LAYOUT/DOS OPEN APPEND
~~~~~~~~~~~~~~~~~~~~~~~~~

MERGE MRG Standard InteraGeneral Vocabulary Operation
Merges two vocabularies and combines them into one.
The current vocabulary is the DESTINATION. Items from the SOURCE will be merged into the destination. Merge lets you first choose the process of selecting the items you desire to bring over from the source to the destination. Then you must choose the mode for handling duplicate names. Pressing <RETURN> selects automatic renaming mode, where all duplicates are kept distinct.
MERGE
Enter the name of the source vocabulary: PATHFILENAME

PATHFILENAME is the name of the source, and may be up to 64 characters long, and may include drive and/or path specifiers, and MUST include a file extension, most probably .PFM, as none is assumed.
SEE ALSO: SELECTIVE SAVE/ /
Merge first asks for the name of the source vocabulary. The current vocabulary is considered the destination, and items from the source will be merged into the destination. The name of the source may be up to 64 characters long, and may include drive and/or path specifiers, and MUST include a file extension, most probably .PFM, as none is assumed. The complete file name must be entered. If you want to cancel MERGE at this point simply rub out the whole line (you can use <CONTRL-ARROW RIGHT> to do this) and press <RETURN>.

Merge then initializes the two files, and this may take a moment.

Merge then asks you to choose the marking mode. Marking is the process of selecting which items you desire to bring over to the destination from the source in the merge. You may cause merge to automatically select all items in the source by pressing <RETURN>. Otherwise, you may press <ESCAPE>, and select individual items, in a tree, similar to the marking method used in SELECTIVE

SAVE. This process DOES NOT actually selective save anything in the source; it ONLY marks items to be brought over to the destination by the merge process. The source vocabulary MUST be written to in order to accomplish this, but this marking will NOT cause any items to be discarded in the source.

If you choose to mark individual items, you are presented with a count of items currently marked, a count of unmarked items, the possibility to LV, LF, or LN, and you may of course type the name of an item (tree) to mark, or press <RETURN> to end the marking phase. The listings (LV, LF, LN) may all be performed on either the source or the destination. When you choose these selections, you may press <RETURN> to cause the listing to be performed on the destination vocab, or <ESCAPE> to list in the source.

After marking is complete, you are asked to confirm the merge by pressing <RETURN> to continue, <ESCAPE> to quit without merging, or any other key to continue individually marking.

After confirming the merge, you are asked to choose the mode for handling duplicate names. You may press <RETURN> to choose automatic renaming mode. This causes ALL duplicate named items to be kept distinct, by adding "!" characters to the beginning of the source name, as necessary, until a unique name is found. This method should be used when combining two different systems, to avoid any problems. It is the safest route.

You may also press <ESCAPE> to choose automatic substitution for the source. This mode is usually used when updating an end user vocabulary to add new features or bug fixes. In this mode, when a duplicate is found, and IF this item is of the same type in both source and destination, the destination item is overwritten with the source item. If the items are NOT the same type, merge defaults to manual handling mode for this item (described next). In order for items of duplicate name to be considered as of the same type, in the case of verbs, it is also mandatory that they use the same number of objects.

You may also press <BACKSPACE> to choose manual handling mode. In this case, each and every duplicate item must be manually resolved in one of four ways.

The first way is to choose automatic renaming for this single item by pressing <RETURN>. Manual handling resumes with subsequent items.

The second way is to choose automatic overwrite of destination with the source (updating method) by pressing <BACKSPACE>.

The third method is to cause this source item to be skipped over entirely, preserving the destination as it is, and causing all source items using this item to instead refer to the destination item.

The forth method is to choose manual renaming of this source item by pressing <SPACE>.

In all cases, the duplicate name display shows you the duplicate item name, and its type in both the source and the destination. If the two items are not of the same type, the "skip the source" and "overwrite the destination" options are NOT available.

Merge processes vocabulary ONLY. Included in the set of vocabulary are file definitions, verb definitions, and nouns complete with value. Specifically excluded from this set are files and their records. When merging two different applications, is may sometimes be necessary to merge the files manually. This may require renaming the .PFM file at the operating system level, and

ALTERing the corresponding file definition shortname to agree.

After all duplicate names are resolved, which occurs during the process of creating all new names in the destination, merge proceeds to the next phase. This is to fixup all definition pointers in the items brought over from the source. When this phase begins, no more interaction is required.

~~~~~~~~~~~~~~~~~~~~~~~~

CREATE WINDOW CWN Standard Verb Screen Operation
Creates a plain window or a window with a border on the screen.
After the creation of the window all display oriented commands will stay
inside the boundaries of the window. You may create as many windows as you
like and select them by using SELECT WINDOW and the desired window number.
Or you can bring a window to the top of a stack of windows with RAISE WINDOW.
DESTROY WINDOW removes a window from the screen.
CREATE WINDOW starting row START ROW starting column START COLUMN ending row
END ROW ending column END COLUMN using border style BORDER NAME number to the
noun NOUN.
BORDER NAME are the standard nouns that define the border style (BORDER NONE,
BORDER DOUBLE COMPOUND, etc). NOUN will contain the number of the new window.
After CREATE WINDOW you must do an ERASE SCREEN to activate the window.
SEE ALSO: SELECT WINDOW/RAISE WINDOW/BORDER DOUBLE COMPOUND
The basic concept of the PRAGMA windowing package is very simple; the current window of the screen (which is a rectangle, with a starting and ending row and column), can be considered to be the whole screen when it is being used. In other words, if you have a window on the screen, and you are DISPLAYing into it, it looks to the program like the window is the whole screen. Everything you DISPLAY stays inside the boundaries of the window. The window will scroll up when you reach the bottom of it. Lines wrap inside the window boundaries. This concept applies totally and uniformly to all display oriented commands.

When inside of a window, the window, for all intents and purposes, is the whole screen.

A good example of this is the standard verb CLEAR SCREEN. If you are inside of a window, when you execute a CLEAR SCREEN, only the window gets cleared. The rest of the screen (if any) stays as it was. Also, the standard verb SAVE SCREEN IMAGE will save only the contents of the window in which it is currently operating. In short, EVERY single display oriented command thinks the current window is the whole screen.

A default window the size of the full screen is automatically created when PRAGMA comes up, and this will be used if you do not create any windows. The full screen (as you see it) is a window, and is thus just like any other window.

CREATING A WINDOW

To create a window, you use the standard verb CREATE WINDOW.

It takes six parameters; the beginning row and column, the ending row and column, the border style and a noun into which the window number is stored.
~~~~~~~~~~~~~~~~~~~~~~~~

The row and column addresses given to CREATE WINDOW are absolute in relation to the physical screen itself. For instance, CREATE WINDOW starting row 10 starting column 15 ending row 20 ending column 65 will create a window starting at row 10 and column 15 of the screen, through row 20 and column 65.

The border style lets you choose whether your window will have no border, a single line border, a double line border, a single line compound or a double line compound border. Compound borders add a column to both sides of the vertical line in order to give a uniform look to the window and not to squash the text against the lines.

The border styles are chosen by entering the standard nouns that describe the border style. These are the standard nouns that you may use:

BORDER DOUBLE (BD)
BORDER DOUBLE COMPOUND (BDC)
BORDER NONE (BN)
BORDER SINGLE (BS)
BORDER SINGLE COMPOUND (BSC)

Each window has a number. When you CREATE WINDOW, the system gives back to you the number of the new window into the noun you specify. You then use this noun to refer to this particular window. The windows number 0 (zero) and 1 are used by PRAGMA.

After you create a window, you can specify the new colors and attribute for it. You must then always do an ERASE SCREEN to fill the window with the chosen colors, and to display the border, if any was chosen.

The cursor will be positioned at the top left corner of the new window. Text written into the window will be in the selected attributes and colors (until they are changed).

WORKING WITH WINDOWS

You may create as many windows on the screen as you like.

When you create a new window, the cursor is moved into that window (to the top left corner). To get back to a window that was previously created, you use SELECT WINDOW. You give it one parameter; the window number. This is how you move the "focus" of PRAGMA from one window to another. The selected window becomes the focus; it becomes the "screen", and all interaction is done, and can only be done, on one window at a time; on the selected window. The selected window remains selected until a new window is selected.

When you create your first window, PRAGMA automatically stores the full background screen into window 1. You can always revert to the full screen and leave a window by issuing the command SELECT WINDOW number 1 in your verb.

When you create many windows, some of them may overlap. When you select a window, if part of it is overlaid by another window, when you display into that covered area of the bottommost (selected) window, this data is written into the window, but it will not be visible on the screen. This data is hidden from view because it is currently underneath another window. If you want to bring the full window up to the top of the stack, to show all of its contents, you use RAISE WINDOW. It also takes a single parameter; the window number. The raised window comes to the top of the pile, all of the window becomes visible, and it may thus now cover part or all of other windows. These covered windows are not deleted or modified in any way; they are just momentarily hidden from view. Selecting a window does not raise it to the top of the stack.

LOWER WINDOW is the opposite of RAISE WINDOW. When you lower a window, it is placed on the bottom of the window stack. If the window being lowered was not at the top of the stack, the currently selected window remains selected. If the window being lowered was at the top of the stack, the mew window at the top of the stack becomes selected. This is different than RAISE WINDOW, which always selects the window being raised.

Normally, in the verb stack, window 1 , the background window, is at the bottom. It's basic purpose is to provide a window to restore the contents of the screen when other windows are destroyed. Window 1 is the same size as the full screen. When you create windows of your own, they are placed in the stack above window 1. And you rarely if ever will need to select or raise window 1. So, it will be there as a copy of the background, as it should be.

When you lower a window, it goes all of the way to the bottom, below even window 1. Since window 1 is full screen sized, this causes a lowered window to disappear. The lowered window is behind window 1, and thus you can't see it. This is a good way to temporarily get rid of a window you don't need for a moment. If you destroy it, you would have to recreate it later, and display into it again. If you lower it, you can make it invisible for the time being, and then simply raise it later when you need it.

The other possibility may be that you want to move a window to the bottom of a stack of your windows, but you want it to still be visible, perhaps partially visible if it is covered somewhat by other windows. To accomplish this, you would lower the window, and then lower window 1 afterwards. Window 1 will then be on the bottom again, and your window will be just above it, at the bottom of the stack of your other windows.

MORE ON WINDOWS

Each window has a state. The state includes the display attribute, foreground and background color, and cursor position. Each window remembers its own state. When you select any window, the character attributes and colors of that window go into affect again, and the cursor moves to the place in the window where it was when you left it (by selecting another window).

Each window also maintains a complete picture of its own contents. When a window is destroyed, the underlying data of the previous window that was overlaid is then restored. This is the main purpose for the full screen default window; it is used to restore the screen when your windows are destroyed.

If you SAVE SCREEN IMAGE on a window that is partially hidden, it makes no difference. You get the

complete and total contents of the window every time.

The default full screen window is always window number 1. This window may also be selected and used as any other. You cannot destroy this window. You may only destroy window created by you. To remove a window from the screen, and thus restore what was on the screen behind it, you use DESTROY WINDOW, and give it the window number.

There is also another window created by the system at start up time. This is window 0 (zero), and it contains the complete image of the physical screen itself. It is also the full size of the screen. This window though contains a composite picture of all windows together. It contains the exact picture of the physical screen. This makes it different than any other window. All other windows contain ONLY the picture of their own contents, and in fact some of these contents may not even be on the screen if part of the window is covered by another.

Thus, window 0 may be considered the physical screen window, and window 1 may be considered the full screen window. Either one may be selected and used as any other window. And neither may be destroyed by you. You may never need these windows, but they are there. An example of a usage of the physical window number 0 is to give you a composite picture of the screen. Perhaps you may create a bunch of windows on the screen. You may then, after you finish creating them, wish to combine them all into ONE big screen. You may do this by selecting window 0, and then performing a SAVE SCREEN IMAGE. The image saved will be the actual total contents of the physical screen. This will be the combined images of all of the windows put together into one (of course, any overlapping areas not on screen will be lost from this composite image). This eliminates the need to join windows together.

The full screen window number 1 can be thought of as a full screen sized window that underlies all of your windows.

There is another verb, DESTROY ALL WINDOWS. This will remove all windows you have created, and the contents of the full screen window number 1 will be restored to the screen. PRAGMA also issues this command when returning to the START MESSAGE.

The basic way to use windows is to create them, select them, use them, and then destroy them when you are done (or let PRAGMA destroy them all when it returns to the START message). Only one window is selected at any given time, and all operations are performed on the selected window. When you create a window, it is also selected automatically. And when you raise a window, it will be selected. Otherwise, you need to explicitly select the window you desire. When you destroy a window, if the window being destroyed is the currently selected window, the topmost window of the stack becomes selected. The window stack is maintained by PRAGMA. When a window is raised, it is put on the top of the stack, and the others get pushed down one. At the bottom of the stack are the two system windows, 0 and 1.

The PRAGMA windowing package works on any terminal supported by PRAGMA, but it will operate slowly on serial terminals (or a terminal like DOSANSI, which may be considered serial for purposes of this discussion), because all windowing functionality must be performed by PRAGMA the hard way. Serial terminals do not have the capability to scroll a window of the screen, for instance, so doing it the hard way is necessary. Under DOS, you should use the memory mapped screens because

they are faster.

~~~~~~~~~~~~~~~~~~~~~~~~

DESTROY WINDOW   DWN Standard Verb Screen Operation
Destroys the specified window and the underlying data of the previous window
that was overlaid is restored. If there are no other windows, the full screen
will be restored.
The default full screen window is always window number 1 and cannot be
destroyed.

DESTROY WINDOW number NOUN

NOUN must contain the number of the window that is to be destroyed.

SEE ALSO: CREATE WINDOW/DESTROY ALL WINDOWS/SELECT WINDOW
~~~~~~~~~~~~~~~~~~~~~~~~~~

SELECT WINDOW   SWN Standard Verb Screen Operation
Selects the specified window and that window becomes the "screen".
The selected window remains selected until a new window is selected.

Selecting a window does NOT raise it to the top of a window stack.

SELECT WINDOW number NOUN

NOUN must contain the number of the window that is to be selected.

SEE ALSO: RAISE WINDOW/CREATE WINDOW/LOWER WINDOW
~~~~~~~~~~~~~~~~~~~~~~~~

RAISE WINDOW   RWN Standard Verb Screen Operation
Raises the specified window to the top of the stack of windows and the
complete window becomes visible.
The raised window may cover part or all of other windows, but the covered
windows are not deleted or modified in any way; they are just momentarily
hidden from view.

RAISE WINDOW number NOUN

NOUN must contain the number of the window that is to be raised.

SEE ALSO: SELECT WINDOW/CREATE WINDOW/LOWER WINDOW
~~~~~~~~~~~~~~~~~~~~~~~~~~

DESTROY ALL WINDOWS DAW Standard Verb Screen Operation
Removes all windows that have been created and the contents of the full screen window (number 1) will be restored to the screen.
This command is also issued when returning to the START message.

DESTROY ALL WINDOWS

SEE ALSO: DESTROY WINDOW/CREATE WINDOW/SELECT WINDOW
~~~~~~~~~~~~~~~~~~~~~~~~~

INSERT LINE ILI Standard Verb Screen Operation
Inserts the specified number of lines BEFORE the line where the cursor currently resides. It does not matter where on the line the cursor is, the new lines inserted will NOT break up a line if the cursor is not at the beginning of it. The newly inserted lines will have the currently selected attributes and will be filled with spaces. The line the cursor is located on and all the following lines are moved downward.
INSERT LINE number of lines NUMBER

NUMBER must be a number or a noun containing a number.

SEE ALSO: DELETE LINE/SCROLL SCREEN DOWN/SCROLL SCREEN UP
~~~~~~~~~~~~~~~~~~~~~~~~~

DELETE LINE DLI Standard Verb Screen Operation
Deletes the specified number of lines, including the line where the cursor is. The following lines are moved up to fill the gap. New blank lines (filled with spaces in the currently selected attributes and colors) fill the bottom lines of the screen that have become empty.
The lines above the cursor do not move.

DELETE LINE number of lines NUMBER

NUMBER must be a number or a noun containing a number.

SEE ALSO: INSERT LINE/SCROLL SCREEN DOWN/SCROLL SCREEN UP
~~~~~~~~~~~~~~~~~~~~~~~~~

SCROLL SCREEN UP SSU Standard Verb Screen Operation
When scrolling the screen up, the entire screen image moves up the number of lines selected and new blank lines, in the currently selected color, enter
~~~~~~~~~~~~~~~~~~~~~~~~~

from the bottom of the screen.
The cursor may be anywhere on the screen and is unaffected by the scroll.

SCROLL SCREEN UP number of lines NUMBER

NUMBER must be a number or a noun containing a number.

SEE ALSO: SCROLL SCREEN DOWN/INSERT LINE/DELETE LINE
~~~~~~~~~~~~~~~~~~~~~~~~~

SCROLL SCREEN DOWN SSD Standard Verb Screen Operation
When scrolling the screen down, the entire screen image moves down the number of lines selected and new blank lines, in the currently selected color, enter from the top of the screen.
The cursor may be anywhere on the screen and is unaffected by the scroll.

SCROLL SCREEN DOWN number of lines NUMBER

NUMBER must be a number or a noun containing a number.

SEE ALSO: SCROLL SCREEN UP/INSERT LINE/DELETE LINE
PRAGMA has the ability to scroll the screen up or down, and to insert and delete lines in the middle of a screen. All of these features are provided via both ANSI commands and standard verbs.

Esc[xxM is the command to delete xx lines. Esc[xxL is the command to insert xx lines. Esc[xxT is the command to scroll the screen down xx lines. Esc[xxS is the command to scroll the screen up xx lines. The corresponding standard verb names are DELETE LINE (DLI), INSERT LINE (ILI), SCROLL SCREEN DOWN (SSD), and SCROLL SCREEN UP (SSU). In all cases, the cursor position is NOT affected by the operation. The cursor will remain in the same absolute location after the operation as it was before it began.

When inserting lines, the new lines are inserted BEFORE the line where the cursor currently resides. It does not matter where on the line the cursor is; the new lines inserted will NOT break up a line if the cursor is not at the beginning of it. The newly inserted lines will have the currently selected attributes and colors, and will be filled with spaces. The line the cursor is located on, and all following lines, are moved downward to open up space on the screen to add the new lines. The bottommost lines will fall from the bottom of the screen. The lines above the cursor do not move.

When deleting lines, the line the cursor is on, and multiple lines if chosen, are deleted. The following lines are moved up to fill the gap. New blank lines (filled with spaces in the currently selected attributes and colors) fill the bottom lines of the screen that have become empty. The cursor does not have to be at the beginning of the line; the entire line where the cursor is located will be deleted, regardless of where the cursor currently resides within that line. The lines above the cursor do not move.
~~~~~~~~~~~~~~~~~~~~~~~~~

When scrolling the screen down, the entire screen image moves down the number of lines selected, bottom lines falling off, and new blank lines (in currently selected color) coming from the top. The cursor does not have to be at the top of the screen in order to scroll down; the cursor may be anywhere, and is unaffected by the scroll.

Scrolling the screen up is similar to scrolling down. The entire image moves, top lines roll off, and new blank lines enter from the bottom, in the currently selected color. The cursor may be anywhere on the screen, and does not have to be on the last line. The cursor does not move.

This new capability of reverse scrolling does NOT affect the current cursor behavior with MOVE CURSOR UP, DISPLAY "_Z", or ANSI relative motion commands with ANSI wrapping enabled. In all of these cases, the cursor will continue to wrap from the top line to the bottom, without scrolling the screen, when you move up from the top line.

Likewise, the new forward scrolling capability does not affect the behavior of MOVE CURSOR DOWN, DISPLAY "_J", and ANSI relative motion commands with ANSI wrap enabled. In all of these cases, the cursor will continue to cause a scrolling if motion downwards from the last line is attempted.

Some terminals have similar scrolling abilities, and if so, these operations can be performed quickly, at the terminal. However, if a certain terminal does not have these abilities, PRAGMA will handle it "the hard way". In this case, PRAGMA will redraw the necessary parts of the screen by sending out line after line of data (except in the case of scrolling up; PRAGMA will temporarily move the cursor to the last line and output a _J, and then restore the cursor position; this is quicker, and PRAGMA expects that all screens can at least do this much...). This may be slow over a modem etc., but at least it will work on any screen.

These capabilities of terminals are described in, of course, TERMINFO.LBS, with new commands. They are "SCRL DN = ", "SCRL UP = ", "INS LINE = ", and "DEL LINE = ". If your terminal does have these capabilities, and if they conform to what PRAGMA expects (as specified above), put the control strings required for your terminal following these commands. If your terminal does not have these abilities, these commands still must be present, and must have a null string given (ex. SCRL DN = ""). If your terminal does not operate in the way PRAGMA expects, you also must null these strings and let PRAGMA do it the hard way; otherwise the terminal and PRAGMA will be at odds as to the contents of the screen.

All of these functions are executed ONE line at a time. Some terminals may allow you to send ONE command to delete multiple lines, for instance. This is not what PRAGMA uses. PRAGMA, when speaking to the terminal, wants to give it a command for a single line operation (ex. delete one line), and PRAGMA will perform that operation multiple times in order to give the intended results. In other words, if you DISPLAY "_[[3M", which means delete 3 lines, PRAGMA will send the string specified in "DEL LINE = " three times to the screen. This was done because not all terminals have multiple line functions; some have only commands that operate on one line. PRAGMA currently uses the least common denominator.

The control string to be used in TERMINFO to cause the screen to scroll up is most probably NOT a simple _J. This will not cause a scroll unless the cursor is on the last line of the screen, and this command must operate regardless of where the cursor is. If your terminal does not have such a command, null this command out, and let PRAGMA do it the hard way (which in this case is not bad at all). The main point is that _J is not the proper command for scroll up.

All of the terminals currently described in TERMINFO.LBS have null strings for all of these new functions. As time permits, each terminal will be tested for compliance with PRAGMA's expectations, and if it has these abilities and they conform, the proper control strings will be added to future versions of TERMINFO.LBS.

The memory mapped screens (for use with DOS) will always have null strings for these features, as in this case they are not needed. Since PRAGMA writes all of screen memory to these type of screens every time any data on screen changes, PRAGMA internally handles all of the details of these functions. Thus, these terminal types are recommended currently for DOS, as they operate very fast.

The DOS ANSI.SYS driver does NOT have these capabilities, and so it will also always have null strings for these functions. When using these terminal types, these functions will always have to be done the hard way, and will thus be slow.

~~~~~~~~~~~~~~~~~~~~~~~~

SET SCREEN SCROLL SSS Standard Verb Screen Operation
Lets you select whether to scroll or wrap the screen when cursor movement extends beyond the top or bottom of the screen.
Scrolling and wrapping can de done in both directions.
The four nouns that can be used with this verb give you all possible combinations. Default is to scroll up when moving below the bottom and to wrap down when moving above the top.
SET SCREEN SCROLL to SCROLL NAME

SCROLL NAME is one of the four standard nouns to be used as the parameter to the verb.
The four standard nouns are SCROLL UP, SCROLL DOWN, SCROLL BOTH and SCROLL NEITHER.
SEE ALSO: SCROLL DOWN/ /
The standard verb SET SCREEN SCROLL lets you select the mode of the screen; either to cause it to scroll or to wrap when cursor movement is attempted beyond the top and bottom boundaries. The possibilities are to either scroll or wrap in either of the two directions; the four nouns SCROLL UP, SCROLL DOWN, SCROLL BOTH and SCROLL NEITHER give you all possible combinations.

The default is to scroll up (when moving beyond the bottom), and to wrap down (when moving beyond the top). This will also be the case after a DELETE ALL WINDOWS, and after going to START.

Each newly created window inherits the scroll/wrap mode currently in effect. This is just like attributes, colors,etc. Each window has its own scroll/wrap mode (this is part of the window state).

The scrolling modes affect both windows and full screens similarly.

SERIAL TERMINALS

A special case exists when using a serial terminal type. It is not known how to stop serial terminals from automatically wrapping when a character is displayed at the very last position of the last line. Indeed, it may not even be possible to inhibit this action. PRAGMA therefore will
~~~~~~~~~~~~~~~~~~~~~~~~

NOT display to this character location at all in certain cases. Namely, if the displaying of a character to this location would cause a scroll, and if that scroll would disrupt the screen. Examples of this are when a full screen sized window is set to SCROLL NEITHER. There are a few other instances as well. In these cases, Pragma will (have to) leave this character cell blank, but the actual character WILL be stored in PRAGMA's internal screen buffer (so that a SAVE SCREEN IMAGE will correctly see it), and the cursor will move forward to the correct location. The only effect will be that this last character cell will remain blank at all times. Also, when using a memory mapped screen, this problem does not occur at all; in that case PRAGMA will display to this last character cell.

~~~~~~~~~~~~~~~~~~~~~~~~~

SCROLL UP   SCU Standard Noun Noun
Parameter to the standard verb SET SCREEN SCROLL (SSS).

The four possible parameters to SET SCREEN SCROLL are SCROLL UP, SCROLL DOWN, SCROLL BOTH and SCROLL NEITHER

SET SCREEN SCROLL to SCROLL UP

SEE ALSO: SET SCREEN SCROLL/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

SCROLL DOWN   SCD Standard Noun Noun
Parameter to the standard verb SET SCREEN SCROLL (SSS).

The four possible parameters to SET SCREEN SCROLL are SCROLL UP, SCROLL DOWN, SCROLL BOTH and SCROLL NEITHER

SET SCREEN SCROLL to SCROLL DOWN

SEE ALSO: SET SCREEN SCROLL/ /
~~~~~~~~~~~~~~~~~~~~~~~~~

SCROLL BOTH   SCB Standard Noun Noun
Parameter to the standard verb SET SCREEN SCROLL (SSS).

The four possible parameters to SET SCREEN SCROLL are SCROLL UP, SCROLL DOWN, SCROLL BOTH and SCROLL NEITHER
~~~~~~~~~~~~~~~~~~~~~~~~~

SET SCREEN SCROLL to SCROLL BOTH

SEE ALSO: SET SCREEN SCROLL/ /
~~~~~~~~~~~~~~~~~~~~~~~~~~

SCROLL NEITHER SCN Standard Noun Noun
Parameter to the standard verb SET SCREEN SCROLL (SSS).

The four possible parameters to SET SCREEN SCROLL are SCROLL UP, SCROLL DOWN, SCROLL BOTH and SCROLL NEITHER

SET SCREEN SCROLL to SCROLL NEITHER

SEE ALSO: SET SCREEN SCROLL/ /
~~~~~~~~~~~~~~~~~~~~~~~~~~

SHOW VERB STACK SVS Debugger Verb Debugging Operation
In the debugger, displays the verb stack.

SHOW VERB STACK

SEE ALSO: DEBUG/LOAD VERB/RUN VERB
~~~~~~~~~~~~~~~~~~~~~~~~~~

BORDER NONE BN Standard Noun Noun
Parameter to the standard verb CREATE WINDOW (CWN).

Using this parameter, CREATE WINDOW displays a window without any borders.

CREATE WINDOW starting row START ROW starting column START COLUMN ending row END ROW ending column END COLUMN using border style BORDER NONE number to the noun NOUN
~~~~~~~~~~~~~~~~~~~~~~~~~~

SEE ALSO: CREATE WINDOW/BORDER SINGLE/BORDER SINGLE COMPOUND
~~~~~~~~~~~~~~~~~~~~~~~~~

BORDER SINGLE   BS   Standard Noun Noun
Parameter to the standard verb CREATE WINDOW (CWN).

Using this parameter, CREATE WINDOW displays a window with a single line border. The window must be at least 3 lines by 3 columns, or the border will not be installed. The border will occupy the outermost character around the window, making the usable size is two characters smaller than specified.
CREATE WINDOW starting row START ROW starting column START COLUMN ending row END ROW ending column END COLUMN using border style BORDER SINGLE number to the noun NOUN

SEE ALSO: CREATE WINDOW/BORDER DOUBLE/BORDER SINGLE COMPOUND
~~~~~~~~~~~~~~~~~~~~~~~~~

BORDER DOUBLE   BD   Standard Noun Noun
Parameter to the standard verb CREATE WINDOW (CWN).

Using this parameter, CREATE WINDOW displays a window with a double line border. The window must be at least 3 lines by 3 columns, or the border will not be installed. The border will occupy the outermost character around the window, making the usable size is two characters smaller than specified.
CREATE WINDOW starting row START ROW starting column START COLUMN ending row END ROW ending column END COLUMN using border style BORDER DOUBLE number to the noun NOUN

SEE ALSO: CREATE WINDOW/BORDER SINGLE/BORDER DOUBLE COMPOUND
~~~~~~~~~~~~~~~~~~~~~~~~~

BORDER SINGLE COUMPOBSC Standard Noun Noun
Parameter to the standard verb CREATE WINDOW (CWN).

Using this parameter, CREATE WINDOW displays a window with a single line border, with the inner and outer columns adjacent to the vertical lines reserved. The window must be at least 3 lines by 7 columns, or the border will not be installed.
CREATE WINDOW starting row START ROW starting column START COLUMN ending row END ROW ending column END COLUMN using border style BORDER SINGLE COMPOUND number to the noun NOUN

SEE ALSO: CREATE WINDOW/BORDER SINGLE/BORDER NONE
~~~~~~~~~~~~~~~~~~~~~~~~~

BORDER DOUBLE COUMPOBDC Standard Noun Noun
Parameter to the standard verb CREATE WINDOW (CWN).

Using this parameter, CREATE WINDOW displays a window with a double line
border, with the inner and outer columns adjacent to the vertical lines
reserved. The window must be at least 3 lines by 7 columns, or the border
will not be installed.
CREATE WINDOW starting row START ROW starting column START COLUMN ending row
END ROW ending column END COLUMN using border style BORDER DOUBLE COMPOUND
number to the noun NOUN

SEE ALSO: CREATE WINDOW/BORDER SINGLE/BORDER NONE
~~~~~~~~~~~~~~~~~~~~~~~~~

ERASE LINE   ELI Standard Verb Screen Operation
Erases lines beginning with the line where the cursor is and proceeds down-
ward. The cursor does not have to be positioned at the beginning of the line.
The complete line where the cursor is located is erased.
After erasing a line or lines the cursor will be in the same place as before
erasing.

ERASE LINE number of lines NUMBER

NUMBER must be a number or a noun containing a number.

SEE ALSO: ERASE TO END OF LINE/ERASE SCREEN/
~~~~~~~~~~~~~~~~~~~~~~~~~

ERASE TO END OF LINEEEL Standard Verb Screen Operation
Erases to the end of the line from the position where the cursor is.

ERASE TO END OF LINE

SEE ALSO: ERASE LINE/ERASE SCREEN/
~~~~~~~~~~~~~~~~~~~~~~~~~

ABORT EXECUTION   AEX Debugger Verb Debugging Operation
Used to quit the debugger after an error occurs during the execution of a verb
(invalid numeric error, for instance). The system will then abort the running
~~~~~~~~~~~~~~~~~~~~~~~~~

verb and return to the operating system. It is necessary to use this command since the debugger terminate key will exit the debugger and resume the program.

ABORT EXECUTION

SEE ALSO: DEBUG/RUN VERB/
~~~~~~~~~~~~~~~~~~~~~~~~~

LOWER WINDOW LWN Standard Verb Screen Operation
Lowers the specified window to the bottom of the stack of windows. If the window being lowered was not at the top of the stack, the currently selected window remains selected. If the window being lowered was at the top of the stack, the new window at the top of the stack becomes selected.
This command usually causes a window to disappear behind window 1, the full screen window.
LOWER WINDOW number NOUN

NOUN must contain the number of the window that is to be lowered.

SEE ALSO: RAISE WINDOW/SELECT WINDOW/CREATE WINDOW
~~~~~~~~~~~~~~~~~~~~~~~~~

EDIT NOUN VALUE ENV Debugger Verb Debugging Operation
Is used in the debugger to change the value of a noun that caused an invalid numeric value to occur during runtime, so that the program may be resumed. Strings must start with the double quote character. Control characters are entered as underline sequences.

EDIT NOUN VALUE the value of NOUN
"string" or number

NOUN is the noun whose value you want to change. "string" or number is the new value for the noun. Strings are entered preceded by a double quote.

SEE ALSO: DEBUG/ABORT EXECUTION/
~~~~~~~~~~~~~~~~~~~~~~~~~

LOGIN NAME LIN Standard TargetNoun
DOS and OS/2: contains an ASCII null value.
UNIX: Contains the login name of the user.

The appropriate value for this noun is set during system startup.
~~~~~~~~~~~~~~~~~~~~~~~~~

DISPLAY the value LOGIN NAME

SEE ALSO: PROCESS ID NUMBER/TERMINAL PORT NAME/OPERATING SYSTEM NAME
~~~~~~~~~~~~~~~~~~~~~~~~~~

TERMINAL PORT NAME TPN Standard TargetNoun
DOS and OS/2: contains "CON" (console).
UNIX: Contains the name of the device node this user is using for the
terminal.
The appropriate value for this noun is set during system startup.
Example: if the screen is on the first serial port (COM1 under DOS), the value
will be "/dev/tty1a".
DISPLAY the value TERMINAL PORT NAME

SEE ALSO: PROCESS ID NUMBER/LOGIN NAME/OPERATING SYSTEM NAME
~~~~~~~~~~~~~~~~~~~~~~~~~~

PROCESS ID NUMBER   PIN Standard TargetNoun
DOS and OS/2: contains 0 (zero).
UNIX: Contains the number assigned by the operating system for the current
PRAGMA session. Each time PRAGMA is started it has a different number.
If multiple users are all running PRAGMA at the same time, each user has
a different number. Each program running in the operating system has a unique
number. The appropriate value for this noun is set during system startup.
DISPLAY the value PROCESS ID NUMBER

SEE ALSO: LOGIN NAME/TERMINAL PORT NAME/OPERATING SYSTEM NAME
~~~~~~~~~~~~~~~~~~~~~~~~~~

OPERATING SYSTEM NAMOSN Standard TargetNoun
DOS: contains "MS-DOS".
OS/2: contains "OS/2".
UNIX: Contains "UNIX".

The appropriate value for this noun is set during system startup.
DISPLAY the value OPERATING SYSTEM NAME
~~~~~~~~~~~~~~~~~~~~~~~~~~

SEE ALSO: TERMINAL NAME/LOGIN NAME/PROCESS ID NUMBER
~~~~~~~~~~~~~~~~~~~~~~~~~

TERMINAL NAME TNM Standard TargetNoun
Contains the value assigned to the environment variable TERM.

Example: "DOSMEM"

The appropriate value for this noun is set during system startup.

DISPLAY the value TERMINAL NAME

SEE ALSO: OPERATING SYSTEM NAME/COLOR SCREEN/SCREEN SPEED
~~~~~~~~~~~~~~~~~~~~~~~~~

SCREEN ROWS SRO Standard TargetNoun
Contains the vertical dimension of the screen, as read from TERMINFO.LBS.

The appropriate value for this noun is set during system startup.

DISPLAY the value SCREEN ROWS

SEE ALSO: SCREEN COLUMNS/TERMINAL NAME/COLOR SCREEN
~~~~~~~~~~~~~~~~~~~~~~~~~

SCREEN COLUMNS SCO Standard TargetNoun
Contains the horizontal dimension of the screen, as read from TERMINFO.LBS.

The appropriate value for this noun is set during system startup.

DISPLAY the value SCREEN COLUMNS

SEE ALSO: SCREEN ROWS/TERMINAL NAME/COLOR SCREEN
~~~~~~~~~~~~~~~~~~~~~~~~~

COLOR SCREEN CSC Standard TargetNoun
Contains a 1 if the screen supports colors, otherwise it will contain a 0 (zero).

The appropriate value for this noun is set during system startup.

DISPLAY the value COLOR SCREEN

SEE ALSO: TERMINAL NAME/SCREEN SPEED/OPERATING SYSTEM NAME

SCREEN SPEED SSP Standard TargetNoun
Contains a 1 if the terminal is memory mapped, otherwise it will contain a 0 (zero).

Memory mapped terminals operate faster than not memory mapped terminals.

The appropriate value for this noun is set during system startup.
DISPLAY the value SCREEN SPEED

SEE ALSO: TERMINAL NAME/OPERATING SYSTEM NAME/COLOR SCREEN

ARC COSINE ACOS Standard Verb Arithmetic Operation
Returns the arc cosine of a number.
If an out of range value is used the result will be zero.

ARC COSINE of the value NUMBER result in NOUN

NUMBER is a number, or a noun containing a number, whose ARC COSINE you wish to calculate. The result of the calculation will be placed into NOUN.

SEE ALSO: COSINE/HYPERBOLIC COSINE/ARC SINE

ARC SINE ASIN Standard Verb Arithmetic Operation
Returns the arc sine of a number.
If an out of range value is used the result will be zero.

ARC SINE of the value NUMBER result in NOUN

NUMBER is a number, or a noun containing a number, whose ARC SINE you wish
to calculate. The result of the calculation will be placed into NOUN.

SEE ALSO: SINE/HYPERBOLIC SINE/ARC COSINE
~~~~~~~~~~~~~~~~~~~~~~~~~

ARC TANGENT   ATAN Standard Verb Arithmetic Operation
Returns the arc tangent of a number.
If an out of range value is used the result will be zero.

ARC TANGENT of the value NUMBER result in NOUN

NUMBER is a number, or a noun containing a number, whose ARC TANGENT you wish
to calculate. The result of the calculation will be placed into NOUN.

SEE ALSO: TANGENT/HYPERBOLIC TANGENT/ARC SINE
~~~~~~~~~~~~~~~~~~~~~~~~~

ROUND UP TO INTEGER RUI Standard Verb Arithmetic Operation
Returns the rounded up value of a number. ROUND UP TO INTEGER always rounds
toward greater numeric values.

Example: ROUND UP TO INTEGER +2.3 = +3
          ROUND UP TO INTEGER -2.3 = -2

ROUND UP TO INTEGER of the value NUMBER result in NOUN

NUMBER is a number, or a noun containing a number, whose ROUND UP TO INTEGER
value you wish to calculate. The result of the calculation will be placed
into NOUN.

SEE ALSO: ROUND DOWN TO INTEGER/ABSOLUTE VALUE/
~~~~~~~~~~~~~~~~~~~~~~~~~

COSINE   COS Standard Verb Arithmetic Operation
Returns the cosine of a number.
If an out of range value is used the result will be zero.
~~~~~~~~~~~~~~~~~~~~~~~~~

COSINE of the value NUMBER result in NOUN

NUMBER is a number, or a noun containing a number, whose COSINE you wish
to calculate. The result of the calculation will be placed into NOUN.

SEE ALSO: ARC COSINE/HYPERBOLIC COSINE/SINE
~~~~~~~~~~~~~~~~~~~~~~~~~

SINE   SIN Standard Verb Arithmetic Operation
Returns the sine of a number.
If an out of range value is used the result will be zero.

SINE of the value NUMBER result in NOUN

NUMBER is a number, or a noun containing a number, whose SINE you wish
to calculate. The result of the calculation will be placed into NOUN.

SEE ALSO: ARC SINE/HYPERBOLIC SINE/COSINE
~~~~~~~~~~~~~~~~~~~~~~~~~

TANGENT   TAN Standard Verb Arithmetic Operation
Returns the tangent of a number.
If an out of range value is used the result will be zero.

TANGENT of the value NUMBER result in NOUN

NUMBER is a number, or a noun containing a number, whose TANGENT you wish
to calculate. The result of the calculation will be placed into NOUN.

SEE ALSO: ARC TANGENT/HYPERBOLIC TANGENT/SINE
~~~~~~~~~~~~~~~~~~~~~~~~~

HYPERBOLIC COSINE   HCOS Standard Verb Arithmetic Operation
Returns the hyperbolic cosine of a number.
If an out of range value is used the result will be zero.

HYPERBOLIC COSINE of the value NUMBER result in NOUN

NUMBER is a number, or a noun containing a number, whose HYPERBOLIC COSINE
you wish to calculate. The result of the calculation will be placed into
~~~~~~~~~~~~~~~~~~~~~~~~~

NOUN.

SEE ALSO: COSINE/ARC COSINE/SINE
~~~~~~~~~~~~~~~~~~~~~~~~~

HYPERBOLIC SINE   HSIN Standard Verb Arithmetic Operation
Returns the hyperbolic sine of a number.
If an out of range value is used the result will be zero.

HYPERBOLIC SINE of the value NUMBER result in NOUN

NUMBER is a number, or a noun containing a number, whose HYPERBOLIC SINE you wish to calculate. The result of the calculation will be placed into NOUN.

SEE ALSO: SINE/ARC SINE/TANGENT
~~~~~~~~~~~~~~~~~~~~~~~~~~

HYPERBOLIC TANGENT HTAN Standard Verb Arithmetic Operation
Returns the hyperbolic tangent of a number.
If an out of range value is used the result will be zero.

HYPERBOLIC TANGENT of the value NUMBER result in NOUN

NUMBER is a number, or a noun containing a number, whose HYPERBOLIC TANGENT you wish to calculate. The result of the calculation will be placed into NOUN.

SEE ALSO: TANGENT/ARC TANGENT/SINE
~~~~~~~~~~~~~~~~~~~~~~~~~

SQUARE ROOT   SQRT Standard Verb Arithmetic Operation
Returns the square root of a number.

SQUARE ROOT of the value NUMBER result in NOUN

NUMBER is a number, or a noun containing a number, whose SQUARE ROOT you wish to calculate. The result of the calculation will be placed into NOUN.

SEE ALSO: RAISE TO A POWER/ /
~~~~~~~~~~~~~~~~~~~~~~~~~~

NATURAL LOGARITHM NLOG Standard Verb Arithmetic Operation
Returns the natural logarithm of a number.

NATURAL LOGARITHM of the value NUMBER result in NOUN

NUMBER is a number, or a noun containing a number, whose NATURAL LOGARITHM you wish to calculate. The result of the calculation will be placed into NOUN.

SEE ALSO: COMMON LOGARITHM/EXPONENTIAL/
~~~~~~~~~~~~~~~~~~~~~~~~~

COMMON LOGARITHM CLOG Standard Verb Arithmetic Operation
Returns the common logarithm of a number.

COMMON LOGARITHM of the value NUMBER result in NOUN

NUMBER is a number, or a noun containing a number, whose COMMON LOGARITHM you wish to calculate. The result of the calculation will be placed into NOUN.

SEE ALSO: NATURAL LOGARITHM/EXPONENTIAL/
~~~~~~~~~~~~~~~~~~~~~~~~~

EXPONENTIAL EXP Standard Verb Arithmetic Operation
Returns the exponential of a number.

EXPONENTIAL of the value NUMBER result in NOUN

NUMBER is a number, or a noun containing a number, whose EXPONENTIAL you wish to calculate. The result of the calculation will be placed into NOUN.

SEE ALSO: NATURAL LOGARITHM/COMMON LOGARITHM/
~~~~~~~~~~~~~~~~~~~~~~~~~

ROUND DOWN TO INTEGERDI Standard Verb Arithmetic Operation
Returns the rounded down value of a number. ROUND DOWN TO INTEGER always rounds toward smaller numeric values.
~~~~~~~~~~~~~~~~~~~~~~~~~

Example: ROUND DOWN TO INTEGER +2.3 = +2
          ROUND DOWN TO INTEGER -2.3 = -3

ROUND DOWN TO INTEGER of the value NUMBER result in NOUN

NUMBER is a number, or a noun containing a number, whose ROUND DOWN TO INTEGER
value you wish to calculate. The result of the calculation will be placed
into NOUN.

SEE ALSO: ROUND UP TO INTEGER/ABSOLUTE VALUE/
~~~~~~~~~~~~~~~~~~~~~~~~~

ABSOLUTE VALUE   ABS Standard Verb Arithmetic Operation
Returns the absolute value of a number.

Example: ABSOLUTE VALUE of +2.3 = 2.3
          ABSOLUTE VALUE of -2.3 = 2.3

ABSOLUTE VALUE of the value NUMBER result in NOUN

NUMBER is a number, or a noun containing a number, whose ABSOLUTE VALUE
you wish to calculate. The result of the calculation will be placed
into NOUN.

SEE ALSO: ROUND UP TO INTEGER/ROUND DOWN TO INTEGER/
~~~~~~~~~~~~~~~~~~~~~~~~~

RAISE TO A POWER   RTP Standard Verb Arithmetic Operation
Raises the value of a number to a power.

Example: RAISE TO A POWER the number 5 to the power of 2 = 25
          RAISE TO A POWER the number 2 to the power of 8 = 256

RAISE TO A POWER the value NUMBER 1 to the power NUMBER 2 result in NOUN

NUMBER 1 is a number, or a noun containing a number, whose value you wish
to RAISE TO A POWER of NUMBER 2. The result of the calculation will be
placed into NOUN.

SEE ALSO: EXPONENTIAL//
~~~~~~~~~~~~~~~~~~~~~~~~~

MODULO   MOD Standard Verb Arithmetic Operation
Returns the modulo of a number. The modulo is the remainder of a number
divided by a divisor.

Example: 10 divided by 7 MODULO = 3
         10 divided by 4 MODULO = 2
         10 divided by 2 MODULO = 0
~~~~~~~~~~~~~~~~~~~~~~~~~

MODULO of the value NUMBER 1 divided by NUMBER 2 result in NOUN

NUMBER 1 is a number, or a noun containing a number, to be divided by NUMBER 2
whose MODULO you wish to calculate. The result of the calculation will be
placed into NOUN.

SEE ALSO: ABSOLUTE VALUE/SPLIT/DIVIDE
~~~~~~~~~~~~~~~~~~~~~~~~~

$INPUT LAYOUT $NL Standard Verb Input/Output Operation
Accepts numerical input with two decimal places (such as dollars and cents)
and a maximum number of designated characters from the keyboard and writes it
in a noun you designate.

LENGTH contains a count of the characters accepted.
$INPUT LAYOUT using layout NUMBER into the noun NOUN

NUMBER is the maximum number of characters that can be entered before the
input terminates. Must be a positive integer.

SEE ALSO: $INPUT/INPUT LAYOUT/INPUT NUMBER
~~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL CREATE PFM XCPF Standard InteraDefinition Operation
Creates automatically an empty .PFM file on the external file path.

This capability is only available if the file has a nonzero mockflag and only
works for PFM. A .DES file is created only for completeness, since it is not
really necessary. PRAGMA also warns you if one or both of these files already
exist, and allows overwrite or abort.
EXTERNAL CREATE PFM FILE for the file named FILE

FILE is the long PRAGMA name of the PFM file taht is to be created.
The empty .PFM file that is created is called SHORTNAME.PFM.

SEE ALSO: CREATE FFM FILE/FILE/
~~~~~~~~~~~~~~~~~~~~~~~~~

CLOSE FILE   CF   Standard Verb File Access Operation
Closes an individual PFM or BTRIEVE file of the internal files.
Any locked records of the user that closes the file are first freed.

Be aware that when closing a file the THIS pointer of that file gets lost.

CLOSE FILE named FILE

FILE is the name of the file that will be closed.
~~~~~~~~~~~~~~~~~~~~~~~~~

SEE ALSO: EXTERNAL CLOSE FILE/CLOSE ALL FILES/
~~~~~~~~~~~~~~~~~~~~~~~~~

RETURN TO OPERATING ROS Standard Verb Execution Flow Control
Causes execution to stop and PRAGMA to exit to the operating system prompt,
bypassing the START message.

RETURN TO OPERATING SYSTEM

SEE ALSO: RETURN TO START/RETURN/REPEAT
~~~~~~~~~~~~~~~~~~~~~~~~~

SEND FILE   SDF Standard Verb Input/Output Operation
Allows any file at the operating system level, either text or binary, to be
sent with a protocol. It uses the communications port that is currently
selected and the port parameters are expected to be set correctly.
Any baud rate, no parity, 8 data bits and 1 stop bit are available. No hand-
shake protocol must be used. Beware that this verb performs a CLOSE ALL FILES
and therefore all THIS pointers will be lost.
SEND FILE filename FILENAME using protocol PROTOCOLNAME

FILENAME may be any valid operating system name, including path and drive.
Wild card characters are not supported in file names.
The following protocols are supported by PROTOCOLNAME:
"XMODEM", "YMODEM" and "KERMIT".
SEE ALSO: SEND/SEND LAYOUT/RECEIVE FILE
SEND FILE (SDF)
RECEIVE FILE (RVF)

The two verbs SEND FILE, SDF and RECEIVE FILE, RVF, allow to send and receive any file at the operating system level, either text or binary. There is no support for modem handling (i.e. dialing) nor terminal emulation.

These verbs use the communications port that is currently selected, and the port parameters are expected to be set correctly. If not, they must be set at the operating system level using whatever means are available. No parity, 8 data bits, and 1 stop bit are required at any baud rate. No handshake protocol is necessary, and may in fact interfere with the operation of the software if used.

Like SYSTEM CALL, these verbs perform a CLOSE ALL FILES, and therefore cause all THIS pointers to be lost.

Two parameters are used with each of these verbs; the first is the file name, and the second is the protocol.

FILENAME

The file name may be any valid operating system name, including path and drive letter. The path and drive letter are only used at the local system. Wild card characters are not supported in file names. In some cases, noted below, the filename is ignored, because the protocol gets the file name from the other connected system in the transmission.

PROTOCOL

The protocol is either "XMODEM", "YMODEM", or "KERMIT". Case is insignificant.

Xmodem and Ymodem protocols are very similar, have a number of variants, and in many cases are confused for each other. There is Xmodem checksum, Xmodem CRC, Xmodem 1K, Ymodem checksum, Ymodem CRC, Ymodem Batch, and perhaps even more than this. It is often hard to tell just what you are going to get when you choose Xmodem or Ymodem in a given software package.

The Xmodem/Ymodem code in PRAGMA attempts, as best it can, to accommodate any and all forms of these two protocols, regardless of which one you have chosen. Your protocol of choice will be the default; it will then attempt to adapt if necessary to what the other side is using. However, this is not always possible, and sometimes you may experience errors. The short answer is to try both; one will usually work better than the other if there are any strange things going on in the software you are communicating with.

When PRAGMA is sending files, using any of the protocols, only one file per command is possible. If the receiving system is using a batch protocol (Ymodem Batch or Kermit), they will see and end of batch indication after PRAGMA completes the sending of the single file. Thus, it is not possible for another system to receive files in batch mode from PRAGMA, even if you attempt to perform many SEND FILE commands in a row. Each SEND FILE command will terminate the batch on the receiving end. PRAGMA can batch receive, but cannot batch send.

When a file is being sent or received, a status window appears in the center of the screen. This window is 11 x 46, appears in the error colors, and is centered. The size and location of the window are not changeable. This window will not interfere in any way with the windows of your application.

The result status of a file transfer can be found in EXTERNAL ECHO and FILE STATUS. If the transfer completes successfully, an "OK" will result; if unsuccessful, a "TRANSMISSION ERROR" will result.

File transfers in progress may be aborted by pressing Ctrl/C twice, which will also cause an entry into the debugger (if enabled).

COMIO

COMIO is necessary when using the DOS version of PRAGMA 4 for any serial communications. In addition, when using Ymodem protocol, the buffer size of COMIO should be configured when it is loaded to be big enough to hold one complete Ymodem communication block. The recommended minimum size of the COMIO buffers in this case is thus 1100.

COMPILING

The code that performs the communications for SEND FILE and RECEIVE FILE is contained in a separate library, COMM.LIB (or Slibcomm.a or similar under UNIX). It must be present when compiling PRAGMA applications. These library files are part of the release diskettes.

TECHNICAL DISCUSSION

For those with a technical bent follows a short description of Xmodem/Ymodem.

The Xmodem in PRAGMA defaults to Xmodem CRC. This is generally preferred, as it has higher error detection than Xmodem checksum. If the other software you are communicating with has multiple forms of Xmodem, and you do not know which one to choose, your best bet is Xmodem CRC.

The Ymodem in PRAGMA defaults to Ymodem Batch. This is also what some people call "True Ymodem". There is another form of Ymodem, without batch capability, that some people also call Xmodem 1K. This is where some of the confusion enters. The Ymodem Batch allows multiple files to be transferred in one session. Ymodem without batch and/or Xmodem 1K do not. You may experience a timeout after the first file is sent if you pair Ymodem Batch with anything else; Ymodem Batch will attempt to send packets announcing more files or end of batch, but the other side, since it does not have batch mode, will have ended and thus will not reply.

The Ymodem in PRAGMA (which is Ymodem Batch) can connect with another Ymodem Batch system, and will also fall back to Xmodem in any of its forms if it sees that come in. However, it may not connect correctly with plain Ymodem. You may see the timeout noted earlier, after the first file has completed.

In general, Ymodem is preferred over Xmodem, as it has some status information that Xmodem does not (file size, file name), but it is more complicated (because of the many forms), and thus you may encounter errors when attempting to communicate with an unknown system. Each system will have to be experimented with until a good match is found.

Kermit protocol is pretty straight forward, as it has the ability built in to reconfigure itself to any level of capability. Any Kermit can adapt to any other Kermit. In Xmodem/Ymodem, this is done outside of the protocol, and is thus not always possible.

The Kermit in PRAGMA is a basic Kermit with checksum; nothing fancy.

Both Ymodem and Kermit know the file name when doing a receive; the file name given as the first

parameter to RECEIVE FILE is ignored in this case. It is only needed for Xmodem. It should be given in all cases however (in case Ymodem switches to Xmodem by itself).

When receiving with Kermit or Xmodem, the receiver does not know the size of the file; only in Ymodem is the file size transmitted to the receiver before the file is sent. The status screen (explained below) will show a blank for the file size if it is not known.

In both Ymodem Batch and Kermit, batch receives are possible. PRAGMA can receive multiple files in one receive command. When using Xmodem, only one file per command is possible.

~~~~~~~~~~~~~~~~~~~~~~~~

RECEIVE FILE RVF Standard Verb Input/Output Operation
Allows any file, either text or binary, to be received with a protocol.
It uses the communications port that is currently selected and the port parameters are expected to be set correctly.
Any baud rate, no parity, 8 data bits and 1 stop bit are available. No handshake protocol must be used. Beware that this verb performs a CLOSE ALL FILES and therefore all THIS pointers will be lost.
RECEIVE FILE filename FILENAME using protocol PROTOCOLNAME

FILENAME may be any valid operating system name, including path and drive.
Wild card characters are not supported in file names.
The following protocols are supported by PROTOCOLNAME:
"XMODEM", "YMODEM" and "KERMIT".
SEE ALSO: RECEIVE/RECEIVE LAYOUT/SEND FILE
SEND FILE (SDF)
RECEIVE FILE (RVF)

The two verbs SEND FILE, SDF and RECEIVE FILE, RVF, allow to send and receive any file at the operating system level, either text or binary. There is no support for modem handling (i.e. dialing) nor terminal emulation.

These verbs use the communications port that is currently selected, and the port parameters are expected to be set correctly. If not, they must be set at the operating system level using whatever means are available. No parity, 8 data bits, and 1 stop bit are required at any baud rate. No handshake protocol is necessary, and may in fact interfere with the operation of the software if used.

Like SYSTEM CALL, these verbs perform a CLOSE ALL FILES, and therefore cause all THIS pointers to be lost.

Two parameters are used with each of these verbs; the first is the file name, and the second is the protocol.

FILENAME
~~~~~~~~~~~~~~~~~~~~~~~~

The file name may be any valid operating system name, including path and drive letter. The path and drive letter are only used at the local system. Wild card characters are not supported in file names. In some cases, noted below, the filename is ignored, because the protocol gets the file name from the other connected system in the transmission.

PROTOCOL

The protocol is either "XMODEM", "YMODEM", or "KERMIT". Case is insignificant.

Xmodem and Ymodem protocols are very similar, have a number of variants, and in many cases are confused for each other. There is Xmodem checksum, Xmodem CRC, Xmodem 1K, Ymodem checksum, Ymodem CRC, Ymodem Batch, and perhaps even more than this. It is often hard to tell just what you are going to get when you choose Xmodem or Ymodem in a given software package.

The Xmodem/Ymodem code in PRAGMA attempts, as best it can, to accommodate any and all forms of these two protocols, regardless of which one you have chosen. Your protocol of choice will be the default; it will then attempt to adapt if necessary to what the other side is using. However, this is not always possible, and sometimes you may experience errors. The short answer is to try both; one will usually work better than the other if there are any strange things going on in the software you are communicating with.

When PRAGMA is sending files, using any of the protocols, only one file per command is possible. If the receiving system is using a batch protocol (Ymodem Batch or Kermit), they will see and end of batch indication after PRAGMA completes the sending of the single file. Thus, it is not possible for another system to receive files in batch mode from PRAGMA, even if you attempt to perform many SEND FILE commands in a row. Each SEND FILE command will terminate the batch on the receiving end. PRAGMA can batch receive, but cannot batch send.

When a file is being sent or received, a status window appears in the center of the screen. This window is 11 x 46, appears in the error colors, and is centered. The size and location of the window are not changeable. This window will not interfere in any way with the windows of your application.

The result status of a file transfer can be found in EXTERNAL ECHO and FILE STATUS. If the transfer completes successfully, an "OK" will result; if unsuccessful, a "TRANSMISSION ERROR" will result.

File transfers in progress may be aborted by pressing Ctrl/C twice, which will also cause an entry into the debugger (if enabled).

COMIO

COMIO is necessary when using the DOS version of PRAGMA 4 for any serial communications. In addition, when using Ymodem protocol, the buffer size of COMIO should be configured when it is

loaded to be big enough to hold one complete Ymodem communication block. The recommended minimum size of the COMIO buffers in this case is thus 1100.

COMPILING

The code that performs the communications for SEND FILE and RECEIVE FILE is contained in a separate library, COMM.LIB (or Slibcomm.a or similar under UNIX). It must be present when compiling PRAGMA applications. These library files are part of the release diskettes.

TECHNICAL DISCUSSION

For those with a technical bent follows a short description of Xmodem/Ymodem.

The Xmodem in PRAGMA defaults to Xmodem CRC. This is generally preferred, as it has higher error detection than Xmodem checksum. If the other software you are communicating with has multiple forms of Xmodem, and you do not know which one to choose, your best bet is Xmodem CRC.

The Ymodem in PRAGMA defaults to Ymodem Batch. This is also what some people call "True Ymodem". There is another form of Ymodem, without batch capability, that some people also call Xmodem 1K. This is where some of the confusion enters. The Ymodem Batch allows multiple files to be transferred in one session. Ymodem without batch and/or Xmodem 1K do not. You may experience a timeout after the first file is sent if you pair Ymodem Batch with anything else; Ymodem Batch will attempt to send packets announcing more files or end of batch, but the other side, since it does not have batch mode, will have ended and thus will not reply.

The Ymodem in PRAGMA (which is Ymodem Batch) can connect with another Ymodem Batch system, and will also fall back to Xmodem in any of its forms if it sees that come in. However, it may not connect correctly with plain Ymodem. You may see the timeout noted earlier, after the first file has completed.

In general, Ymodem is preferred over Xmodem, as it has some status information that Xmodem does not (file size, file name), but it is more complicated (because of the many forms), and thus you may encounter errors when attempting to communicate with an unknown system. Each system will have to be experimented with until a good match is found.

Kermit protocol is pretty straight forward, as it has the ability built in to reconfigure itself to any level of capability. Any Kermit can adapt to any other Kermit. In Xmodem/Ymodem, this is done outside of the protocol, and is thus not always possible.

The Kermit in PRAGMA is a basic Kermit with checksum; nothing fancy.

Both Ymodem and Kermit know the file name when doing a receive; the file name given as the first parameter to RECEIVE FILE is ignored in this case. It is only needed for Xmodem. It should be given in all cases however (in case Ymodem switches to Xmodem by itself).

When receiving with Kermit or Xmodem, the receiver does not know the size of the file; only in Ymodem is the file size transmitted to the receiver before the file is sent. The status screen (explained below) will show a blank for the file size if it is not known.

In both Ymodem Batch and Kermit, batch receives are possible. PRAGMA can receive multiple files in one receive command. When using Xmodem, only one file per command is possible.

~~~~~~~~~~~~~~~~~~~~~~~~

READ BINARY FILE RBF Standard Verb Operating System Operation
Reads a binary file (or any file) on the operating system and writes it to a noun.
READ BINARY FILE opens the specified file, reads the entire contents of it and places this into the destination noun as a binary value. The file is then closed.
The length of the data read is placed into the standard noun length.
READ BINARY FILE filename FILENAME into the noun BINARY

FILENAME is an expression or noun containing the name (and optional path) of the operating system file you want to read into the noun BINARY.

SEE ALSO: WRITE BINARY FILE/DOS OPEN READ/DOS OPEN CREATE
~~~~~~~~~~~~~~~~~~~~~~~~~~

WRITE BINARY FILE WBF Standard Verb Operating System Operation
Writes binary data in a noun to a file in the operating system.

WRITE BINARY FILE opens the specified file, truncates its length to zero and then writes the entire contents of the source noun to it. The file is then closed.

WRITE BINARY FILE filename FILENAME from the noun BINARY

FILENAME is an expression or noun containing the name (and optional path) of the operating system file you want to write to the binary data from the noun BINARY.

SEE ALSO: READ BINARY FILE/DOS OPEN READ/DOS OPEN CREATE
~~~~~~~~~~~~~~~~~~~~~~~~~~

EXTERNAL CLOSE FILE XCF Standard Verb External File Access Operation
Closes an individual PFM or BTRIEVE file of the external files.
Any locked records of the user that closes the file are first freed.

Be aware that when closing a file the THIS pointer of that file gets lost.

EXTERNAL CLOSE FILE mamed FILE

FILE is the name of the file that will be closed.
~~~~~~~~~~~~~~~~~~~~~~~~~~

SEE ALSO: CLOSE FILE/CLOSE ALL FILES/
~~~~~~~~~~~~~~~~~~~~~~~~~

 Version 4.40 a  Standard Noun Noun
This records is used only to store the version number that is displayed in the PRAGMA system of the online utility.

SEE ALSO: / /
~~~~~~~~~~~~~~~~~~~~~~~~~