FNSNAP Functions

Modified and augmented by George Tisdale
Based on a collection of functions from Emphasys Software

Table of Contents

Library Functions

Disclaimer

The following library functions, written in the Business Rules Language have been contributed by various users of the Business Rules Language and provided to the BRGroup for free distribution. You may use these functions freely in your program, but you are solely responsible for the proper implementation and resulting conditions. The BRGroup and the original donor make no warrantees whatsoever with respect to the appropriateness of these functions in any application to which the ultimate user may employ them.

Overview of Libraries and Functions

Function

A FUNCTION in Business Rules is an extremely useful tool. Using a function you can perform a series of similar tasks on one or more variables and return the changed variable, a result of the interaction of those variables on one another, or a combination of both. Generally a function is created by using a DEF clause. If the function only needs one line of code then there are no additional lines, the function stands on its own. More likely, however, is a multilined function, these also start with a DEF clause followed by various BR statements and end with an FNEND statement.

The lines of code between the DEF and the FNEND statements follow the same rules as "normal" BR code with a few exceptions relating to variables passed in to the function. If a variable is passed into a function it can be "by reference" or "direct".

By Reference

A by reference function variable is sent to the function, but is not returned by the function. That is it is left unchanged in the main program. (There are exceptions to this also if the function resides in the same program as the calling program.)

Direct

A direct function variable is sent to the function and whatever changes the function makes on the variable are returned to the calling program. Direct variables are designated in the DEF statement with a leading ampersand "&". Matrices passed to a function are ALWAYS direct. In practice what is really occurring is that direct variables are the actual variable in memory and the function makes changes to the original variable, rather than a referred value as is the case in a referred variable.

Matrix changes

As mentioned above a matrix is a direct variable. It can be re dimensioned, changed in many ways and then used by the calling program in it's changed state. The only time when this will not happen is when the matrix is non-existent in the function call. A non-existent matrix is one that has been marked as an "Optional" variable in the DEF statement and no matrix was named in the program call to the function. The non-existent matrix can not then be re-dimensioned because it dies not exist. If it is desired to re-dimension a matrix and have it passed out of the function then it is necessary to use a "dummy" matrix passed in so that the function does not use a NULL by default.

Library Function

A function can either be a part of the program that is loaded, similar to a sub-routine, or it can be a library function that is either present in the program or located in another named program. A library function definition differs from a function definition because it includes the word library between the DEF and the name of the function. The program code for a library function is otherwise the same as a function.

In order to use a library function the library and the library function name must be designated in the program or library that calls the library function. This is done with a "LIBRARY" statement.

```
LIBRARY "E:\WB\vol002\fnsnap.dll":FNPRINTBOX,FNDRAWBOX
```

The above statement says that the library functions FNPRINTBOX and FNDRAWBOX are located in the BR program "FNSNAP.dll" that has been named with a dll suffix, rather than a BR or WB suffix, located in the "wb\vol002" directory of drive E:. Note that BR automatically assigns a BR or WB suffix to programs when they are initially created, but there is no reason that a BR program cannot have any suffix that you assign to it.

Library Variables

Any variable used with a program or library is common to all the functions, sub-routines and main programs within that loaded program or library. This means that if, in library SAMPLE.br, library function FNFIRST uses a variable "X" to count how many times a routine is processed and it turns out that that number is 10, and library function FNSECOND, also included in SAMPLE.br, uses the variable "X" which it expects is initialized to zero because that function has not used it before, unexpected results will occur because the value of "X" is now 10, not zero. If FNSECOND were located in a different library, then the value of "X" set by FNFIRST

would not be visible to it and FNSECOND would not use the value of 10, but might use a value previously set by FNSECOND. As a result it is imperative in writing functions that variables that are expected to be clear are initialized within the library function to be clear before being used in the library function.

Library Sub-routines

Sub-routines in program are very useful. They can be used from different parts of the program without the necessity of duplicating code. They can also be used to keep complex manipulations out of the main logic of a program to make it easier to understand. Sub-routines can be used in functions, library functions and libraries as well. Two library functions residing in the same library can each use the same sub-routine, located outside of the DEF / FNEND boundaries. This is true as long as the sub-routine is located within the same library as the library function(s) making use of it.

Library Status

Once a library has been created and it is being named in a program it can be loaded as a "RESIDENT", library, a "CURRENT" library or a "RELEASE" library.

Resident Library

A resident library once loaded remains in memory and can be called by any program that is subsequently loaded. It retains its variable values from program to program. It is only removed with a clear command or by exiting the BR session.

```
00100 library resident "E:\wb\vol002\getdates.br":FNDATE
```

Current Library

The default load of a library is a current library. It is active while the program that loads it is in memory and retains its variables only while that program is active. A current library, because it terminates when a program ends cannot call a resident library.

```
00100 library "E:\wb\vol002\getdates.br":FNDATE
```

Release Library

A library that is designated as release only maintains its variables while a call to it is active. As soon as the function reaches the FNEND statement and passes information back to the calling function or routine the variables are cleared. A

released library, since it does not occupy memory when not active cannot call a current or a resident library.

```
00100 library release "E:\wb\vol002\getdates.br":FNDATE
```

FILE

File Management

FNBLDSORT - build a sort control file and execute the sort

Creates a sort control file with FILE RECORD and MASK statements. The sort is then executed creating the sorted file in either PD 3, BH 4 or Record out format. The sort control file is then deleted. The user's local TEMP directory is used for the work space but the files in and out files can be located anywhere either as relative or absolute paths.

```
FNBLDSORT(INNM$,OUTNM$,ABR$,MASK$,MAT
     RECORD$,INDIR$,INDRV$,OUTDIR$,OUTDRV$)
```

Functions used:

None

Variables:

INNM$	File name that is being sorted. The file name can contain the path either relative (without a drive letter) or absolute (with a drive letter). If no path is included FNBLDSORT will look in the current directory for the file.
OUTNM$	File name that is to be created by the sort routine. The file name can include a path just as it could with the INNM$.
ABR$	Type of out[ut file to be created A=Address using the PD 3 format B=Address using the BH4 format R=Record out sort.
MASK$	The file mask that should be used to designate what positions and format the sort should use in creating the sort. The mask statement should NOT include the word MASK, but the remainder of the mask should follow the parameters detailed in the BRManual under the SORT Control facility. These include multiple sort positions and types in the start position, field length, field format and ascending or decending as in "11,5,C,A" or "11,5,C,A,21,5,PD D". Only one MASK statement is allowed, but the statement may include multiple parameter sets.

The following elements are optional and can be omitted from the function call.

MAT RECORD$ Multiple RECORD statements can be used and can contain a mix of INCLUDE and OMIT statements. Each Include or Omit statement must be in a separate element of the RECORD$ array and NOT include the word RECORD. A statement may look something like the following:

```
01000 DIM RECORD$(0)*100
01010 MAT RECORD$(1) !:
     RECORD$(1)='O,49,1,C," "," "'
01010 MAT RECORD$(2) !:
     RECORD$(2)='I,23,5,C,"aaaaa","zzzzz"'
```

Notice the use of single quotes and double quotes to avoid the necessity of using multiple double quote mars in order to get quotes included in the RECORD$ array elements.

INDIR$ This parameter is included for compatibility with the sort utility. The parameter should be left blank (not, included in the function call).

INDRV$ This parameter is included for compatibility with the sort utility. The parameter should be left blank (not, included in the function call).

OUTDIR$ This parameter is included for compatibility with the sort utility. The parameter should be left blank (not, included in the function call).

OUTDRV$ This parameter is included for compatibility with the sort utility. The parameter should be left blank (not, included in the function call).

Comments:

A very useful function where sorts are used. PArticularly helpful in eliminating the procedure files common in older program. Using this function chain to procedure files to create and execute sorts can be included in the primary program much as FNINDEX is used.

FNFIL - create a file number that will increment by 1 each time a batch is created

Create a file number and batch number for a file that will increment by 1 each time a new batch is created. Used to create sequential file names such as transmittal files to banks or taxing authorities where it important to maintain a record of what has been sent and make sure that an existing transmittal file is not overwritten.

Returns an available file number to be used in creating a file and the sequence number to be used in opening the file.

Functions used:

```
FNFIL(FILLOC$*100,FILNM$,&FILBATCH;SUFX$)
```

Functions used:

Variables:

FILLOC$	Location either relative of specific of the path into which the file should be placed. The path should end with a "\". If one is not present one will be added to the path.
FILNM$	The first few letters of the file name to be used. generally this is tow to three letters to identify the specific reason for the file
&FILBATCH	
SUFX$	

Comments:

An example might be

```
01010 let x=fnfil("E:\wb\efile\","MAW",fseq,"txt")
01020 open
     #x:"name=E:\wb\efile\MAW"&cnvrt$("PIC(#####)",fseq)&".txt,recl=3200,
     replace",display,output
```

==========

FNFILENAME$*80 - Provide a file name

```
FNFILENAME$*80(;NAME$*80)
```

Description

Functions used:

FNOK
FNWIN
FNCLSWIN

Variables:

NAME$	An optional name to display for a file

Comments:

An older version of FNGETFILE$ and FNPUTFILE$

FNFILEOK - Check version number

Checks version number and file length and returns the file version number. FNFILEOK returns a variable indicating file status

```
FNFILEOK(NUMBER,NAME$,LENGTH,FILE_VERSION;&OLD_VERSION)
```

Returned values for FNFILEOK

1	New file was created
2	Record length was adjusted, but file versions match
3	File versions do not match. The existing version is higher than FILE_VERSION
4	File versions do not match. The existing version is lower than FILE_VERSION

Functions used:
FNSIZE (part of FNSNAP.dll)

Variables:

NUMBER	file number being processed
NAME$	name of file being processed
LENGTH	record length the file SHOULD be
FILE_VERSION	current file version
OLD_VERSION	file version of existing file

Comments:
If the LENGTH is greater than the length of the old file the file will be expanded to the new file size. If the actual file is greater than the requested length the record length will NOT be changed.

The function returns a value depending on what is found

FNFILESIZE - count number of records in a file

Returns the number of records in a named file - unnecessary, use LREC

```
FNFILESIZE(FILENAME$*66)
```

Returns the number of records in a named file

Functions used

Variables:

FILENAME$ NAme of file to be analyzed

Comments:

This is an unnecessary function because BR returns an LREC for any open file which is the number of records in the file. However, because the function works on a file that has NOT been opened you may find it helpful

FNGETFILE$ - Return the name and path of an existing file

Returns the path and name of a file that has been picked using FILEDIALOG.exe

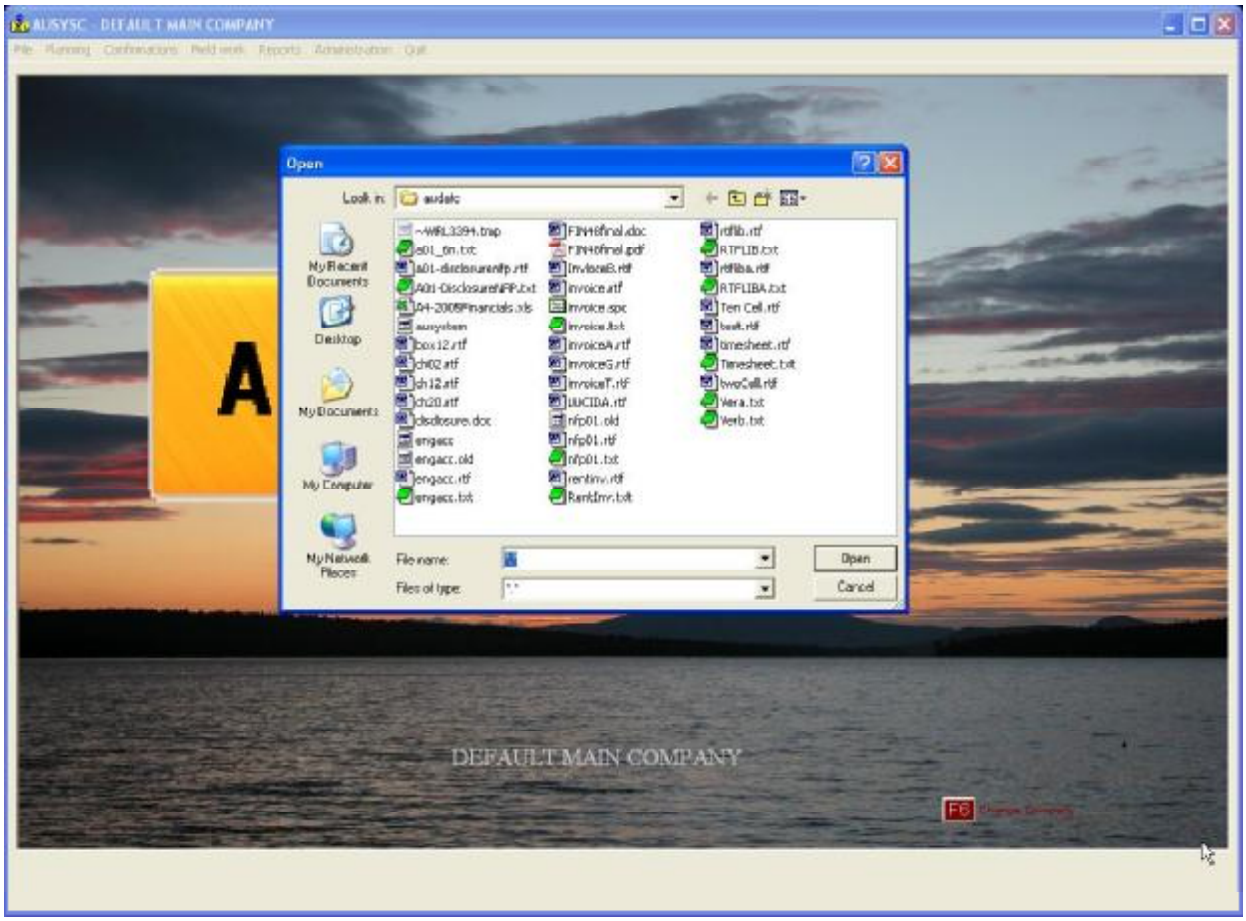


Figure: SNAP0010.ptf

```
FNGETFILE(LOOKIN$,LOOKFOR$)
```

Functions used:

Functions used:
None

Variables:

LOOKIN$	Name of the path for which to display the selection API dialog window.
LOOKFOR$	The type of file to be included in the file look up.

Comments:

This is a BR access to the Windows API call for retrieving any file from the disk drive.

FNGETFILENAME$*80 - Create a file name from a seed

```
FNGETFILENAME$*80(SEED$;D)
```

Description
Displays a window and request a file name. Optionally a suggested name can be displayed.

Functions used:
None

Variables:

SEED$	Portion of a file name. The suggested returned file name such as AP
D	Flag to indicate that the seed name should be suffixed by the current date in MMDD format

Comments:

Useful in creating export files or other temporary files.

FNINDEX - Build an Index file and check for duplicates

Builds an index file for the specified file.If Duplicate keys are not allowed displays a message box allowing deletion of duplicates or ignoring the error

```
FNINDEX(FLNR,FLNM$*40,KFNM$*40,KS$,KL$,DUPS,KFMSG$*200)
```

Functions used:

Functions used:
FNPRINT_FILE
FNDIALOG

Variables:

FLNR	Number assigned to file when and if it is opened by FNINDEX
FLNM$	File name including path if necessary
KFNM$	Key file name including path if necessary
KS$	Key starting position(s)
KL$	Key length(s)
DUPS	If true (non-zero) then duplicate records are allowed, if false (zero) then duplicate keys cause a trapped error that allows display and printing of the duplicates. The condition can also be ignored with a Continue or the duplicate keys can be deleted.
KFMSG$	Message to display on the command console while index is being built

Comments:

FNNEXTFIL$ - returns the file name of the next sequential file in a given location

Similar purpose to FNFIL above, but this returns the name of the file to be used.

```
FNNEXTFIL$*100(NFIL$,NPATH$*100)
```

Functions used:

Variables:

NFIL$	The first few characters of the file name to be created
NAPTH$	The path where the file is to be created. If NPATH$ does not end with a "\" one will be added.

Comments:
Used by FNOPEN in the reprint series of functions (See printing).

==========

FNPUTFILE$ - Return the name and path of an file to be created, replaced or appended

Functions used:

Returns the path and name of a file that has been picked using FILEDIALOG.exe

```
FNPUTFILE(LOOKIN$,LOOKFOR$)
```

Functions used:
None

Variables:

LOOKIN$	Name of the path for which to display the selection API dialog window.
LOOKFOR$	The type of file to be included in the file look up.

Comments:

This is a BR access to the Windows API call for returning the name of any file from the disk drive which we want to modify in some way.

FNSIZE - Set or correct a file size

Changes file size of referenced file. Will only increase file size, will not shrink file.

```
FNSIZE(FLNM$,FLLEN) ! Function to change file size
```

Functions used:
None

Variables:

FLNM$	name of file being processed
FLLEN	length of new file

Comments:

Useful in conjunction with a routine to check file versions. The function will check the names file and if its length is less than the number passed the file will be copied to a new file with the -D and -S parameters which remove deleted records and increase the record length of each record to the -S length.

FNUPDATE_VERSION - Change a file version and reconfigure layout

Functions used:

Converts a file from one version to another based on passed FORM statements

```
FNUPDATE_VERSION(FILENAME$,DIRNAME$,MAT VERSIONS,DIR
     COPY$,OLDFORM$*500,NEWFORM$*500;LASTREC,DELETE_LASTREC)
```

Functions used:

Variables:

FILENAME$	File name of file to be converted without any path name
DIRNAME$	Directory where FILENAME$ exists
MAT VERSIONS	a 2 by 3 matrix old file information is on line 1 new file information is on line 2. Columns for each are column 1 version number. Column 2 size of the string matrix needed (Mat A$) Column 3 size of the numeric matrix needed (Mat A)
DIRCOPY$	Work directory for update process. If directory does not exist it will be created.
OLDFORM$	Compiled FORM statement for the old version of the file
NEWFORM$	Compiled FORM statement for the new version of the file
LASTREC	If TRUE and DELETE_LASTREC is FALSE then a record 1 in the format L9 is created with the number of records in the file
DELETE_LASTREC	If TRUE then no record 1 with number of records in the file is created in the new version file

Comments:
This routine is quite old and does not allow the rearranging of fields that is possible using the newer techniques developed by Gabriel Bakker.

Data management

FNSEQ - Return the next sequence for a key field

Returns a number that is the next sequence number for a keyed file key that uses a number following the primary key to keep the records unique.

```
FNSEQ(FILNR,FILKEY$,FILFRM$)
```

Functions used

Variables:

FILNR	Number of already open internal, keyed file
FILKEY$	The key that will be sequenced
FILFRM$	Form statement for reading the key and suffix ex. "FORM pos 7,c 10,n 3"

Comments:

Useful in adding a sequence number in a keyed file so that duplicate keys are not created.

FNTYPE - Move the contents of one text file into another open file

Moves the contents of a named file into an existing open display file.

```
FNTYPE(INFILE$*100,OUTFILE)
```

Functions used:
FNPROGRESS (This progress bar can be disabled in the function without damage to the function)

Variables:

INFILE$	File name including path of any display file, the contents of which are to be moved to an open print file.
OUTFILE	File number of an open print file. The print file should have been opened using EOL=NONE to avoid CR and LF being inserted where they are not wanted.

Comments:
Excellent for moving a graphic into a print file where the graphic or other image is greater than 32,000 characters long. Can also be used to merge a PCL macro or form with a printed page.

Form statements

FNCFORM$ - Create a Condensed Compiled Form Statement

Takes a string of field specifications and creates a compiled FORM variable combing repetitive specifications into a bracketed multiple specification n order to fit the statement within the size limitation for a compiled form statement.

```
FNCFORM$*2000(ACF$*2000)
```

ACF$ is a string of field specifications separated by commas. Both are dimensioned to more than are allowable.

Functions used:

FNCF$(ACF$)

Comments:
A string of "C 5,C 5,C 5,C 5" will be returned by FNCF$ as "4*C 5" The function FNCFORM$ takes this revised specification, adds a "FORM " to the front and compiles it into a compiled format variable.

FNCF$ - Process a field specification string for use in FNCFORM$

Takes a string of field specifications and creates condensed specifications string by combing repetitive specifications into a bracketed multiple specification.

```
FNCF$*2000(ACF$*2000)
```

ACF$ is a string of field specifications separated by commas. Both are dimensioned to more than are allowable.

Functions used:

None

Comments:
A string of "C 5,C 5,C 5,C 5" will be returned by FNCF$ as "4*C 5"

Screen Processing

Buttons, messages and dialogs

FNBUTTON - Add button on the button bar

Creates a button on the button bar. Buttons are created left to right and can be removed with FNCLRBUTTON

```
FNBUTTON(BUTTON_TEXT$,FK;BTN)
```

Functions used:

Variables:

BUTTON_TEXT$	Text to display on button. Must be less than 10 characters
FK	Function key value to return when the button is pressed
BTN	Button number if an existing button is to be changed. If this parameter is omitted the next button position will be used.

Comments:

FNCHECK - in connection with a radio dot or check box returns a 1 if checked

Used to process the elements of a radio dot list or check box is to determine wether the element has been checked (true) or not (false)

```
FNCHECK(L$*100)
```

Functions used:

Variables:

L$	The label description being processed for an element of a radio list or check b ox list

Comments:
This function is intended to be used by other functions see for example FNOPTIONS and FNOPTIONS$

==========

FNCHECK$ - places or strips ^ from an element

Based on the results of FNCHECK or a default parameter this function will add or remove the ^ from a description that indicates if it has been checked

Returns the label modified to either contain or be free of a leading ^.

```
FNCHECK$*100(L$*100,L)
```

Functions used:

Variables:

L$	The label description to be modified if L is true
L	A flag indicating whether the ^ is to be appended to the front of a label, or stripped from it

Comments:

Used as a part of FNOPTIONS and FNOPTIONS$

==========

FNCLRBUTTON - removes a button from the button bar

Removes a button from the button bar if it was created using FNBUTTON

```
FNCLRBUTTON(;BTN)
```

Functions used

Variables:

BTN	The number of the button to be cleared. If blank the last, highest number, button will be cleared. If equal to 99 all buttons will be cleared.

Comments:

Use in connection with FNBUTTON to display and remove buttons left to right on the button bar.

FNDIALOG$*40 - Display dialog box and return selected text

Displays a dialog box with up to three options and specifiable text. See also FNDLG

```
FNDIALOG$*40(SROW$,SCOL$,DBWIDTH,TXTSTR$*900,OPT1$*40,OPT2$*40,OPT3$*40,R
     EMOVE,DFLTOPT,DISPANYKEY,KEYWAIT) !:
```

Functions used

Variables:

Comments:

Text does not align very well in the box when using proportional fonts. This function can be called from FNDLG that uses an unformatted BR file to supply text and button labels. Both functions have been updated to utilize the external utility from David Blankenship.

FNDLG - Display a dialog box from data in a file

Creates a dialog box from a DAT file prepared by DIALOGMN.BR

```
FNDLG(DIALOG_DAT,DLNR;DISPANYKEY,KEYWAIT,SUFFIX$*300)
```

Functions used
FNDIALOG

Variables:
DIALOG_DAT
DLNR
DISPANYKEY
KEYWAIT
SUFFIX$

Comments:

This is a carry over from earlier versions and does not align proportional text very well. If possible use MSGBOX instead.

THe function has been updated to use the message box utility from David Blankenship if RADIOCHK.exe is in the VOL002 directory

FNHELP - open a tip box associated with a screen using a text file

Searches a specified text file for an anchor point, then displays a record after that point specified by HFLD. Used for displaying pop up help windows based on the input field where the cursor is located

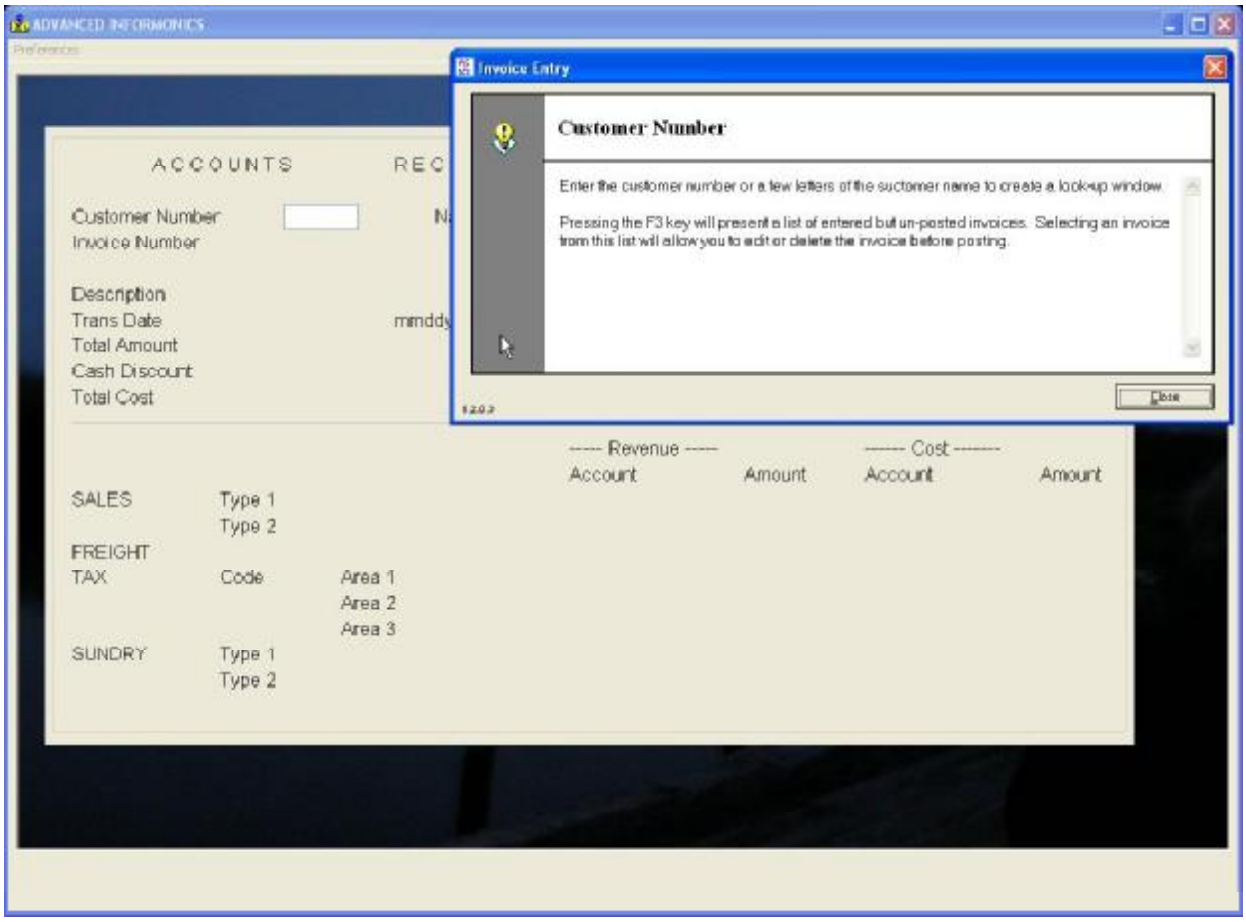


Figure: SNAP0007.ptf

```
FNHELP(HPATH$*60,HFILE$*20,HBASE$,HFLD,HROW,HCOL;HTITLE$*80)
```

Functions used:

Variables:

HPATH$	Path where the help file is located
HFILE$	File name within HPATH$ for the help file

HFLD	Field number, usually set by CURFLD except in GRIDs it should be set by CURCOL.
HROW	Current row position of cursor, used to help in the positioning of the help window. Usually set by CURROW
HCOL	Current column of cursor, used to help in the positioning of the help window. Usually set by CURCOL
HTITLE$	An optional title to appear in the bar at the top of the help window.

Comments:
Very flexible and easy to implement help system. I uses David Blankenship's HELPTIPS.exe utility.

==========

FNHELPTIP - uses David Blankenship utility to display a help record

Displays a record from a text file in a pop up window. Used in the BR system by the ERRORS routine to display the description of an error number.

```
FNHELPTIP(PROGPATH$*100,TEXTFILE$*50,TITLE$*50,RECORD,HROW,HCOL;NO_WAIT)
```

Functions used:

Variables:

PROGPATH$	Path where text file is located.
TEXTFILE$	Name of text file within the specified path
TITLE$	Name to be displayed at the top of the pop up window
RECORD	The record number within the text file to display as the text of the message.
HROW	A positioning variable generally set by CURROW
HCOL	A positioning variable generally set by CURCOL
NO_WAIT	Ignored

Comments:
If HROW and HCOL are zero then the window id positioned in the center of the screen.

==========

FNOK - Pop-up "OK" question

```
FNOK
```

Description
Displays a dialog box with OK yes or NO. If yes is returned FNOK is true else it is false

Functions used

Variables:

Comments:
MSGBOX is probably a better option with current programs

FNOPTIONS - creates a radio dot selection pop up

Displays a pop up radio dot selection window. If David Blankenship's utility RADIOCHEK.exe is present that will be used. If not present then a BR generated selection window will be generated.

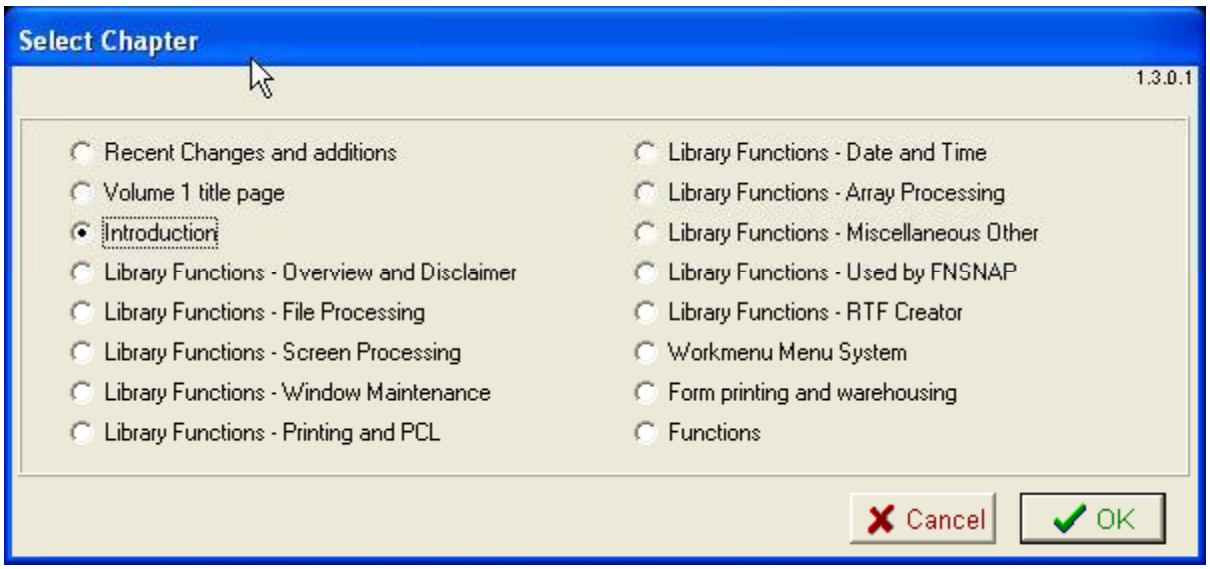


Figure: SNAP0005.ptf

```
FNOPTIONS(MAT O$;DEFAULT,TITLE$*100,MESSAGE$*1000,WAITTIME,SROW,SCOL)
```

Functions used:

FNRADIOCHK

Variables:

Mat O$	Matrix containing the descriptions for each line in the list
DEFAULT	A number indicating which radio dot item should carry the dot when the box is displayed
TITLE$	A title to appear across the top of the list box.
MESSAGE$	A message to appear in a separate part of the dialog box explaining the choices if appropriate.
WAITTIME	Number of seconds that the list should be displayed before accepting whatever is checked and continuing.
SROW	Positioning parameter generally set by CURROW
SCOL	Positioning parameter generally set by CURCOL

Comments:

==========

FNOPTIONS$ - creates a check box selection pop up

Similar to FNOPTIONS, but this function displays the information in the form of a multiple selection check box list.

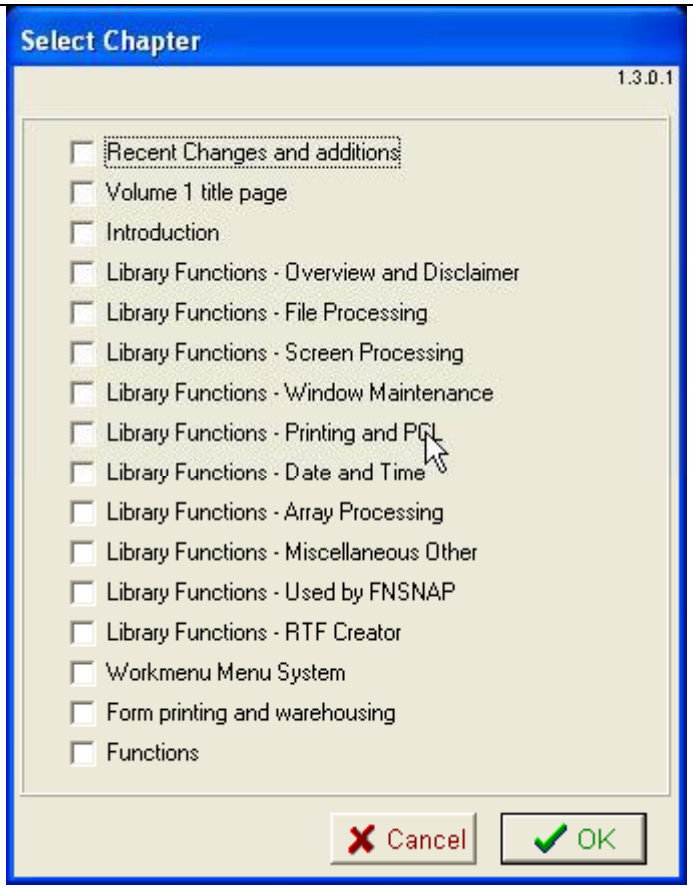


Figure: SNAP0006.ptf

```
FNOPTIONS$*100(MAT
     O$;DEFAULT$*100,TITLE$*100,MESSAGE$*1000,WAITTIME,NONE)
```

Functions used:
FNRADIOCHK

Variables:

DEFAULT	A number indicating which radio dot item should carry the dot when the box is displayed
TITLE$	A title to appear across the top of the list box.
MESSAGE$	A message to appear in a separate part of the dialog box explaining the choices if appropriate.
WAITTIME	Number of seconds that the list should be displayed before accepting whatever is checked and continuing.
NONE	A flag that will allow no items to be selected, otherwise the check list may not be exited without at least one selection being made.Positioning parameter generally set by CURROW

Comments:

==========

FNPFKEYLINE - Creates a hot field string of function key options

See also FNWINBUTTONS for 4.17+

Prints a function key message line in an open window or child window at the row specified. Function keys displayed are hot and return the Fkey value. Fkey references can be hot text or optionally buttons.

Figure: SNAP0009.ptf

```
FNPFKEYLINE(ROW,TXT$*80;FKWIN)
```

Functions used

Variables:

ROW	The row number of the window on which the line should be displayed. Negative number indicate that many rows UP from the bottom of the window. Zero "0" indicates the very bottom of the window. All messages will be right justified within the window.
TXT$	Text to be displayed. Function keys should be designated with a leading carat ^ and trailing double space such as "^Esc End"
FKWIN	Window number in which the line should be displayed

Comments:

Valid function keys are numbers beginning with "^F" such a ^F9, or abbreviations for keys including ^PgUp ^PgDn ^Esc and ^Enter. These are not case sensitive.

FNPFKEY - Prints a function key message

```
FNPFKEY(R,C,F$,TXT$*78) !:
```

Description
Prints a function key message on window zero

Functions used

Variables:

R	Row number
C	Column number
F$	Function key to be displayed
TXT$	Message to be displayed next to the function key

Comments:
Displays only one function key message at the row and column of window #0 specified. FNPFKEYLINE is more flexible and recommended.

FNRADIOCHK$ - display a radio/checkbox with a set of options

Used by FNOPTONS and FNOPTIONS$ and FNDLG to display radio dot list, check box list, or a dialog box. Makes use of David Blankenship's RADIOCHK.exe utility.

```
FNRADIOCHK$*100(CAPTION$*80,INFILE$*60,LEFT,TOP,ALLOW,DEFAULT$*100,TYPE$,
     LOCATE,NOCOLS,COLWIDTH,WAITTIME;TEXTSTRING$*2400)
```

Functions used:

Variables:

CAPTION$	Title for the top bar
INFILE$	File name to be processed
LEFT	Position form left of screen to display in pixels
TOP	Position from the top of the screen in pixels
ALLOW	allows no responses if true, requires at least one if false
DEFAULT$	A string of 0 and 1 where 0 is not checked and 1 is checked
TYPE$	R for Radio C for Check box
LOCATE	Record number in file to use as data for a dialog box

NOCOLS	Number of columns to display
COLWIDTH	Width of columns or buttons. If not provided the spacing will be automatic based on the length of text provided
WAITTIME$	Number of seconds to wait before returning the default answer.
TEXTSTRING$	Text to display in a dialog box if not using text from a file.

Comments:

==========

FNRADNUM - Return the option selected in a radio dot list

```
FNRADNUM(MAT V$)
```

Description
Function determines which in a group of radio buttons was selected and returns the element number

Functions used

Variables:

Comments:

FNTIMEOUT - Display timeout message

```
FNTIMEOUT(;SECONDS)
```

Description
Displays a message that input has timed out and waits for a keystroke to reactivate the program.

Functions used

Variables:

Comments:

FNWINBUTTONS - print one or more buttons on a screen in a designated window

Similar to and a replacement of FNPFKEYLINE. Prints one or more buttons, right justified in a designated window either BROWs down from the top of the window if BROPW is positive of BROWS up from the bottom of the window if BROW is zero or negative.

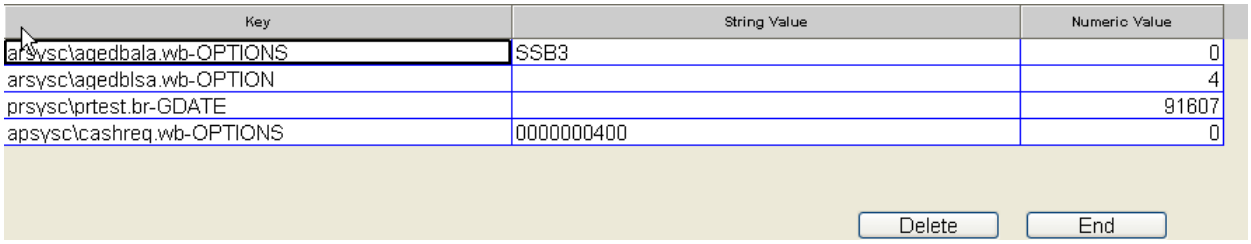


Figure: SNAP0002.ptf

```
DEF LIBRARY FNWINBUTTONS(BROW,BTEXT$*100,BWIN)
```

Functions used:

FNWINROWCOL

Variables:

BROW	The row number within a window on which to place the buttons. If 0 or negative the row is up from the bottom of the window, positive is down from the top.
BTEXT$	The text to display within the buttons. Each button is designated with a ^followed by FX: where X is the fkey value to be returned when the button is pushed. FX can also be PGUP, PGDN or ESC. The^FX: is followed by the text to appear within each button. Al buttons are dimensioned to the longest string provided for any button.
BWIN	The window number of the window within which the buttons should appear.

Comments:

Can only be used in GUI ON mode

==========

String Manipulation

FNDECRYPT$ - Decrypt FNENCRYPT$

Undoes what FNENCRYPT$ does

```
FNDECRYPT$(PW$)
```

Functions used

Variables:

PW$	The encrypted password from FNENCRYPT$. This is usually a stored value to be compared with an entered value

Comments:
This is a simple encryption routine only meant to hide a value from a casual observer, not a dedicated hacker. A more robust encryption routine is available with George Tisdale's WORKMENU.br menuing system.

FNENCRYPT$ - Simple encryption

```
FNENCRYPT$(PW$) !:
```

Description
Simple encryption routine to ward off SNOOPS, NOT serious hackers

Functions used

Variables:

Comments:
This is a simple encryption routine only meant to hide a value from a casual observer, not a dedicated hacker.

FNFKEY - Converts an FKEY value greater than 1000

Returns a function key value. If the value would be greater than 1000 then only the right tree places are used to determine the number. Useful in converting button FKEYs to hot text fkeys

```
FNFKEY(AKEY)
```

Functions used

Variables:

AKEY	An FKEY value to be reduced by 1000 and returned as the value of FKEY. If AKEY is less than 1000 the value of AKEY will be returned

Comments:

Use in conjunction with FNPFKEYLINE

FNNUM$ - Convert number to string

```
FNNUM$(NAMT,DCML,LNGTH) !:
```

Description
Converts a number to a character string with fixed decimal places. CNVRT$ is a better option.

Functions used

Variables:

Comments:

FNPHONE$ - Convert number to (###) ###-####

Formats a 10 digit number into a telephone number formatted string

```
FNPHONE$(X)
```

Functions used

Variables:

X	Numeric telephone number including a leading "1"

Comments:

FNPROPER$*60 - Convert to Proper Case

Converts a string into an initial capital title or name case

```
FNPROPER$*60(A_IN$*60) !:
```

Functions used

Variables:

A_IN$	String that is to be converted to initial capitals

Comments:
The function uses a lot of SREP$ statements, consequently the string passed should be short enough to not cause a string overflow. About 500 characters is a reasonable limit.

Other

FNAUTO - 1 if last field exit was automatic

Returns a 1 (true) if the last field was exited with an automatic exit Attribute E or X
LASTFIELD is the field # that was current

```
FNAUTO(LASTFLD) !:
```

Functions used

Variables:
NONE

Comments:

FNCLKBUF - Clear keyboard buffer

Clears the keyboard buffer

```
FNCLKBUF ! Clear the keyboard buffer
```

Functions used:
None

Variables:
None

Comments:

FNERRTRAP - Trapped Error Processing

Red Screen error trapping routine. Displays error, program line and number to call. Also logs the error to a log file by workstation and creates an email message

```
FNERRTRAP(EPROG$*50,ELINE,EERR,ECOUNT,EVARIABLE$,&ECURFLD,EMENU$)
```

Functions used

Variables:

EPROG$	Program name where the error was generated
ELINE	Line number where the error occurred
EERR	Error number that occurred

ECOUNT	Variable count if appropriate to the error
EVARIABLE$	Name of the variable, if appropriate where the error occurred
ECURFLD	Current field where the error occurred if in full screen processing
EMENU$	The menu to which the program should return if the error can not be resolved
EMENUSEQ$	The menu Sequence to which the program should return if the error can not be resolved.

Comments:

Requires FNEMAILFILE and EmailBlaster or EmailMonitor

FNINIT - Initialize variables in FNSNAP Library

Initiates the variables for the FNSNAP library. Be careful not to use more than once in a program

```
FNINIT(;SYSDIR$,SYS$)
```

Functions used

Variables:
NONE

Comments:
Initializes variables for the older FNSNAP tools. If run in the middle of a program the variables will be reset to the initial values and may cause problems. Most new functions being written should NOT use this function.

FNPRINTSCREEN - stuff the keyboard to generate a print screen

Programmatically controls the keyboard to do the equivalent of Ctrl-P to issue a print screen

```
FNPRINTSCREEN
```

Functions used:

Variables:
None

Comments:

==========

FNZERO - set to number if zero

```
FNZERO(V,DV) ! SET Variable equal to the Default Variable if zero !:
```

Description

Screen input and display

FNMOD - returns the column number of a cell in a grid

Replaced by CURCOL in 4.17. Returns the column number of the current cell

```
FNMOD(CEL,COLS)
```

Functions used:

Variables:

CEL	The cell number where the cursor is located
COLS	The number of columns in the grid

Comments:

==========

FNPARSERES - returns screen resolution and BR window size for a session

Converts

```
FNPARSERES(W$,MAT SCRNRES,MAT WINRES,MAT CONRES)
```

Functions used:

Variables:

W$	The workstation ID being queried
MAT SCRNRES	Two element array of current terminal rows and columns in pixels
MAT WINRES	Five element matrix
1	0 is maximized ("M-") 1 is windowed ("A-")
2	rows in pixels of window
3	columns in pixels of window
4	row position of upper left corner in pixels of window
5	columns position of upper left corner in pixels of window
MAT CONRES	Five element matrix
1	0 is maximized ("M-") 1 is windowed ("A-")
2	rows in pixels of window
3	columns in pixels of window
4	row position of upper left corner in pixels of window
5	columns position of upper left corner in pixels of window

Comments:
Uses Steve Koger's RESOLUTION.exe utility

==========

FNPROGRESS - Progress bar

```
FNPROGRESS(&PCT_WINDEV,PCT_TOTAL,PCT_DONE;SR$,CAPTION$*55)
```

Description
Displays a progress bar that expands based on numbers passed to the function

Functions used

Variables:

Comments:

FNSCREEN - 24 x 80 screen display for screen painter

```
FNSCREEN(SCRNO;SCREENFILE,MAT SCRATR$,MAT SCREEN$,MAT INWRK$,MAT
     INFLDA$,MAT INWRKH$,NOPAINT) ! Retrieve and display screen
```

Description
Displays a generated screen in the full screen window 0. Screen was created using SCREENMN in a 23x80 format.

This function has been significantly changed by NEWSCREEN.dll

Functions used

Variables:

Comments:

FNTEXTBOX - creates a text box with word wrap

Displays and allows input from a windows text box with text

```
FNTEXTBOX$*4000(&TEXTWIN,SROW,SCOL,ROWS,COLS,TLEN,PARENT,TEXT$*4000;BORDE
     R,TKEY$)
```

Functions used:

Variables:

TEXTWIN	
SROW	Starting row within the parent window where the upper left corner should appear
SCOL	Starting column number within the parent window where the upper left corner should appear
ROWS	Number of rows the text box should cover
COLS	Number of Columns the text box should cover
TLEN	Allowable length of the text
PARENT	Parent window number
TEXT$	Text to display. Modified text will be returned as the value of the function

BORDER	Zero for no border or any other number to create a single line border
TKEY$	Function key to return if the window is to be marked as hot

Comments:

==========

FNWINSCRN - paints a screen in a window

```
FNWINSCRN(SFIL,SCRNO,WINNO,WINLIN,WINLEN,MAT SINFLDA$,MATSHELP$;DISPLAY)
     !:
```

Description
Displays a generated screen in an open child window of specified size. The generated screen was created using SCREENMN

This function has been significantly changed by NEWSCRN.DLL

Functions used

Variables:

Comments:

FNWINROWCOL - in GUI mode returns rows and columns of a window

Must be in GUI ON mode to use this function. Returns the number of rows and number of columns in the specified window. the values returned are actually one shorter than the actual size of the window.

```
FNWINROWCOL(WINNO,&WROWS,&WCOLS)
```

Functions used:
FNWINSIZE

Variables:

WINNO	Window number for which information is requested
WROWS	Number of rows less one of the requested window
WCOLS	Number of columns less on of the requested window

Comments:

==========

FNWINSIZE - in GUI mode creates arrays holding all window sizes

Returns arrays carrying dimensions for all open windows

```
FNWINSIZE(MAT S_WINNO,MAT S_SROW,MAT S_SCOL,MAT S_EROW,MAT S_ECOL,MAT
    S_ROWS,MAT S_COLS,MAT S_PARENT)
```

Functions used:

Variables:

MAT S_WINNO	Array is dynamically populated with window number information
MAT S_SROW	Array is dynamically populated with starting row number
MAT S_SCOL	Array is dynamically populated with starting column number
MAT S_EROW	Array is dynamically populated with ending row number
MAT S_ECOL	Array is dynamically populated with ending column number
MAT S_ROWS	Array is dynamically populated with the number of rows in the window
MAT S_COLS	Array is dynamically populated with the number of columns in the window
MAT S_PARENT	Array is dynamically populated with the number of the parent window

Comments:
this is the working function for FNWINROWCOL

==========

Window Maintenance

List and Grid

FNLISTSPEC$*50 - Create a window for a list/grid box

```
FNLISTSPEC$*50(&LISTWIN,SR,SC,LROWS,LCOLS,AROWS,MAT H$,MAT W,MAT
     F$;HTEXT$*100,G$)
```

Description
Creates a window that contains either a list box or a grid

Functions used FNWINHEAD

Variables:

LISTWIN	
SR	Startin row posito of the upper left corner of the grid
SC	Starting column position of the upper left corner of the grid
LROWS	Number of rows that should be provided for the grid
LCOLS	Number of columns wide that shouldbe provided for the grid
AROWS	Number of extra rows above the size of the grid to include in the window to allow for buttons or other informatin at the bottom.
MAT H$	Header title array information for the grid
MAT W	Width array specificatin for the grid
MAT F$	Format specificatin array for the list
HTEXT$	Text to display in a header bar. If text is included than the listbox/grid aill contain a blue header. If HTEXT is blank no header will appear.
G$	Blank for a list box "GRID" to create a grid

Comments:

Child windows

FNWINHEAD - Print the top bar to a window

Prints a windows look alike bar at the top of a window using graphics that are stored in the ICONS directory directly below the BR root.

Figure: SNAP0001.ptf

```
FNWINHEAD(HWIN,HTEXT$*100,HLEN)
```

Description
Creates the top row of a window in windows look-a-like mode with a clickable X and a title

Functions used

Variables:

HWIN	Window number that the top bar is to be placed over
HTEXT$	Message of title that shuld appear printed in white within the top bar
HLEN	Length of the bar, which should be the dsame as the width of the window referenced in HWIN

Comments:
The bar will be another window which will be a child of the underlying parent window. When the parent is closed the child will automatically close also.

FNWINDEV - Query FNSNAP for last window opened

Function transfers from FNSAP to the calling program the window number of the last window opened using FNWIN. This practice is being replace in 4.17 becasue multiple windows can be opened an closed at will and does not require the strict opening of windows in a specific order.

```
FNWINDEV
```

Description
Returns the value of the currently open window that has been opened by FNWIN

Functions used

Variables:

Comments:

Printing and PCL

Font Management

FNFONT$*30 - Create a PCL font string

```
FNFONT$*30(SYMBOL_SET$,PROPORTIONAL,CHR_PER_INCH,STYLE$,WEIGHT$,TYPEFACE$
    )
```

Description
Creates an HP 5 PCL font string from certain parameters

Functions used

Variables:

Comments:

FNLOADFONT$*50 - Loads a PCL font to printer

```
FNLOADFONT$*50(NUMBER$,FONTCALL$*50;FONT$*100,OUTFILE)
```

Description
Moves a downloadable font into an open display file for printing and returns the font calling string to the program. If the font file does not exist or is invalid the font string is still returned to the program.

Functions used

Variables:

Comments:

GENERAL

Reprinting Reports

FNCLEANLOG - part of FNREPRINT used to remove out of date reports

Reviews a log file of reports available for reprinting. If no destroy date has been entered a default of 30 days is entered. If a report delete date has expired FNCLEANLOG will delete the report and update the log to indicate the date of deletion. If a report is marked as deleted and the deletion occurred more than 7 days ago then the report log entry will be removed.

```
FNCLEANLOG(;REPORTLOG)
```

Functions used:

Variables:

REPORTLOG	Optional file number for a report log to be processed. Generally omitted
LOGNAME$	Not entered her, LOGNAME$ has previously been stored as a variable in the library by another function.

Comments:
This function is generally run by pressing F4 in the reprint reports list box.

==========

FNOPEN - create a log file for saved reports

Opens a display file in a specified directory. The file name is determined as a sequence number with the leading characters specified in FLNM$. The name of the open file and the number of the open file are returned to the calling program, ready for creating a RAW print file.

```
FNOPEN(&FLNM$,&FLPATH$;PRINTDESC$*80,LLEN,PRINTTYPE$,SAVE_DAYS)
```

Functions used:

Variables:

FLNM$	The leading few characters of the file name to be opened. The function will complete the file name with a sequence number for that type file in the specified path.

FLPATH$	The path, either absolute or relative, where the print file should be created
PRINTDESC$	A description of the file to appear in the REPORTLOG reprint dialog listing
LLEN	The length of each line in the display file or zero (0) if EOL=NONE should be used in the open statement
PRINTTYPE$	If omitted "ALL" will be entered meaning the report can be printed to an NWP printer using preview. Other options are "DIRECT" and "MATRIX" if a specific printer type is required due to character string that are not compatible with NWP.
SAVE_DAYS	The number of days a report should be retained for reprinting. If zero (0) is entered no destruction date will be entered, but the first time that FNCLEANLOG is run after the report creation a destroy date of 30 days will be substituted.

Comments:
To save a report for a long time enter a destroy date number of days significantly in the future such as 365 or 500

==========

FNMENUACCESS - used in connection with FNREPRINT to determine user permission to reprint a report

If WORKMENU is being used then FNMENUACCESS checks the permission files to determine if the user has permission to create the report. If no permission exists to create the report then it cannot be reprinted and is excluded from the detail list of reports available for reprint.

```
FNMENUACCESS(MNAME$*10,MSEQ$*3,MPGM$*50)
```

Functions used:

Variables:

MNAME$	NAme of the menu from which a report was created
MSEQ$	Sequence number of the menu from which the report was created
MPGM$	The program call in WORKMENU that allowed the report to be created

Comments:

This function is used by FNREPRINT to determine user rights for reprinting a report. The variables are all obtained from REPROTLOG.fil and passed to this function to determine whether access should be granted for reprinting.

==========

FNPRINT - prints a saved report opened using FNOPEN

Issued immediately after the CLOSE statement of a file opened using FNOPEN to send the stored print file to a specified printer. Print specification substitutions for the specified printer are performed during this print process for the RAW print file.

```
FNPRINT(FILNM$*100,PRINTER$*50)
```

Functions used:

Variables:

FILNM$	The name and path of the stored RAW file to be printed.
PRINTER$	A normal BR printer designation. Can be a substitutable printer type such as \\10 or a specific printer such as \\server01\hplaser

Comments:

FILNM$ is generally shown as filpath$&filnm$, the two variables returned to the calling program by the FNOPEN function

==========

FNREPRINT - displays a list of available saved reports and prints selected report

Displays a list of report quantity by month created and allows selection of a month. The user is then presented with a list of the reports created that month for which permission exists for reprinting.

Figure: SNAP0003.ptf

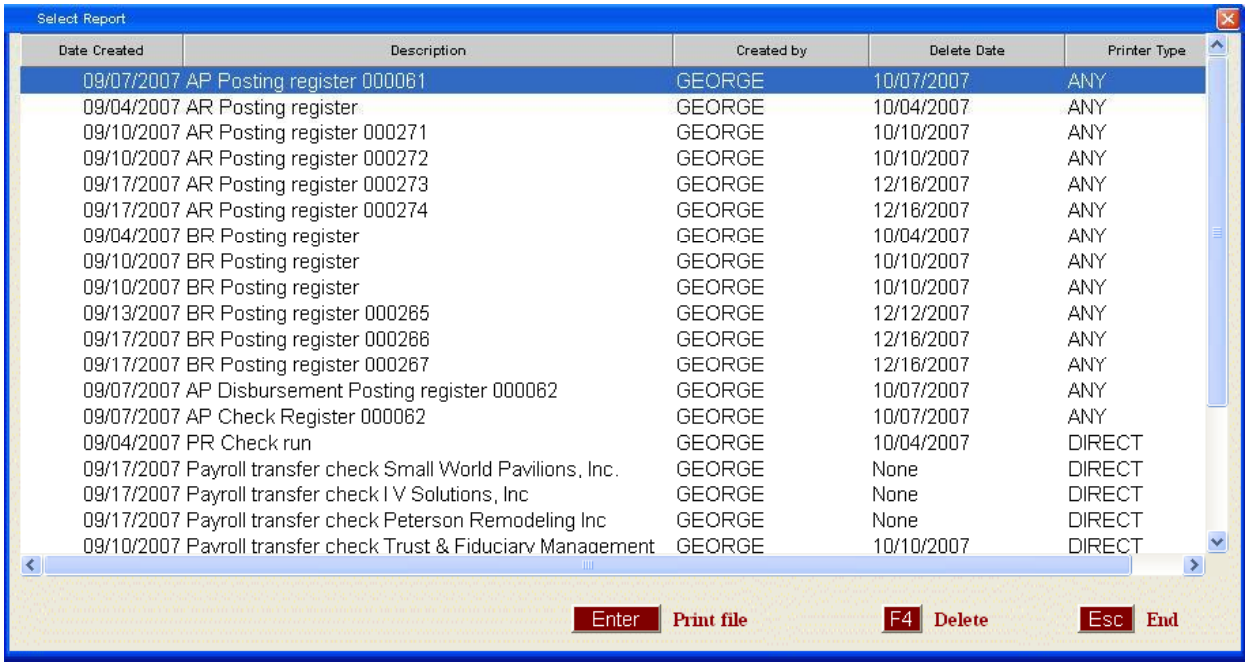


Figure: SNAP0004.ptf

```
FNREPRINT(;ALL,LOGNAME$*100,LOGKEY$*100)
```

Functions used:

Variables:

ALL	If true shows all entries regardless of security rights
LOGNAME$	Name of the log file if other than REPORTLOG.fil
LOGKEY$	Name of the index file if other than REPORTLOG.idx

Comments:

A program can be created that includes only this function to allow menu access to reprinting

==========

PCL and NWP formatting

Bar Codes and addresses

FNBARCODEM - Prints postal bar code to a MATRIX printer

```
FNBARCODEM(ODEV,ZIP$;INDENT)
```

Prints a postal bar code on a matrix printer

Functions used:
NONE

Variables:

ODEV	File number of open print job
ZIP$	Zip Code to be translated to Postal Net
INDENT	Default is 10 characters. If other than the default is desired enter the character position of the start of the bar code.

Comments:

FNCODE3OF9 - Creates 3 of 9 Bar code in PCL

Prints a bar code in 3 of 9 format

```
FNCODE3OF9(PRINTFILE,V,H,TEXT$*30,PRNTXT$;HEIGHT,CHECKD) !:
```

Functions used:
FNPRINTBOX

Variables:
PRINTFILE
V
H
TEXT$
PRNTXT$
HEIGHT
CHECKED

Comments:

FNCODEUPC - Creates UPC bar code in PCL

```
FNCODEUPC(PRINTFILE,V,H,TEXT$*30;HEIGHT) !:
```

Description
Prints a bar code in UPC format

Functions used

Variables:

Comments:

FNENVELOPE - Prints an envelope with return address and Postal Bar Code

Prints an envelope on a laser printer with postal bar code and a return address if a specific overlay file exists (this will be changed in the future to make the return address an option

```
FNENVELOPE(PRTFILE,DATAFILE,SIZE$;SUPRET,MAT INNAMES$,NOLBLS,NOCLOSE)
```

Functions used:

FNPOSTNET
FNPRINTBOX
FNTYPE

Variables:

PRTFILE	Number of open print file to which envelope will be printed
DATAFILE	Number of the file that contains the graphic for the return address
SIZE$	Code indicating the envelope size to be printed
SUPRET	Return address is suppressed if this is TRUE, if FALSE return address and graphic are printed
MAT INNAMES$	Matrix containing the name and address to be printed
NOLBLS	Number of copies of the printed envelope to be printed
NOCLOSE	If this is TRUE then the print file is to be left open. If false the default is to close the print file after printing the envelope.

Comments:

FNGETZIP - extracts a zip code from an address line

Searches then end of the passed string to obtain a valid zip code. If one is found then the zip code excluding any dash is returned as the value of the function

```
FNGETZIP$(ADD$*50)
```

Functions used:

Variables:

ADD$	A right trimmed string that carries a zip code at the right hand end.

Comments:

==========

FNLABEL - prints a 3 1/3 x 4 laser label on 3 x 2 stock

Prints a mailing label including postal zip bar code on a 3x4 6 to a sheet laser printed label

```
FNLABEL(FILNUM,MAT FADD$,MAT TADD$;START,NUMBER)
```

Functions used:
FNPRINTBOX
FNDRAWBOX
FNGETZIP
FNPOSTNET

Variables:

FILENUM	File number of already open display or print file
MAT FADD$	From Address matrix. Array of three elements
MAT TADD$	To address matrix. Array of 3 or 4 elements
START	Starting number on the label sheet containing 6 3 x 4 labels
NUMBER	Quantity of labels to prepare

Comments:

==========

FNPOSTNET - Prints the Postal Bar Code created by FNPOSTNET$

```
FNPOSTNET(PRINTFILE,V,H,TEXT$*20) !:
```

Description
Print postal bar code to a laser printer in PCL format

Functions used:
FNPOSTNET

Variables:

PRINTFILE	Number of open print file to which the bar code will be printed
V	Vertical position of the upper left corner of the bar code in inches
H	Horizontal position of the upper left corner of the bar code in inches
TEXT$	Postal zip code to be translated to a bar code

Comments:
Works in PCL and NWP modes

FNPOSTNET$*4000 - Creates a postal bar code in PCL

```
FNPOSTNET$*4000(TEXT$*20) !:
```

Description
Creates a postal bar code in PCL format

Functions used:
FNPRINTBOX

Variables:

TEXT$	The postal bar code in a string variable.

Comments:
If the string variable cannot be converted into a valid postal zip code FNPOSTNET will return a blank.

Forms and formatting

FNDRAWBOX - Prints a four sided shaded box on PCL

```
FNDRAWBOX(PRINTFILE,VP,HP,VL,HL,WEIGHT;FILL)
```

Description
PCL5 code to print a box with outline and shading to an HP compatible laser printer

Functions used

Variables:

Comments:

FNGREYBAR - Creates the overlay used in FNGREYBAR$

```
FNGREYBAR(PRINTFILE,V,H,BV,BH,SHADE,HEAD,BAR)
```

Description
Creates the gray bar PCL code used by FNGREYBAR$

Functions used FNPRINTBOX

Variables:

Comments:

FNGREYBAR$ - Overlays a printout with gray bar effect

```
FNGREYBAR$(MACRO,PRINTFILE,V,H,BV,BH,SHADE,HEAD,BAR)
```

Description
Creates a PCL5 macro that simulates green bar paper and returns the macro call

Functions used FNGREYBAR

Variables:

MACRO	Macro number to assign
PRINTFILE	Number of open print file
V	Upper left corner of paper in inches (usually 0)
H	Upper left corner of the paper in inches (usually 0 but could be 0.5 to allow for notebook holes).
BV	Vertical height in inches of the area to be covered with gray bars
BH	Horizontal width in inches of each the area to be covered by gray bars
SHADE	Depth of shade of the bars in multiples of 10 form 0 to 100 (recommend 20 or 30)
HEAD	Size in inches of the blank space at the top for title and other header information
BAR	height on inches of the gray bars

Comments:

FNMAKEPCL - converts an HP6l saved file into a file for PCL overlay

Processes a display file created by print through an HP6L print driver to a file. FNMAKEPCL removes the characters necessary to prepare the file for being a MACRO overlay or a part of a continuous print job.

```
FNMAKEPCL(INFILE$*100,OUTFILE$*100)
```

Functions used:

Variables:

INFILE$	The name of the saved HP6L print file to be converted
OUTFILE$	The name of the file to be created as a result of the conversion

Comments:

==========

FNPRINTBOX - Creates a PCL line and positions formatted text in PCL

```
FNPRINTBOX(PRINTFILE,V,H,BV,BH,SHADE;TV,TH,TEXT$*6000,CPI,FONT$*40) !:
```

Description
PCL5 code to print a line and optionally formatted text to an HP compatible lase printer

Functions used

Variables:

V	Vertical position of upper left corner of print area in inches
H	Horizontal position of upper right corner of print area in inches
BV	Vertical depth of the print area below V in inches
BH	Horizontal width of the print area to the right of H
SHADE	Index for gray shading in multiples of 10 from 0 (white) to 100 (black)
TV	Vertical position of text to print below V in inches
TH	Horizontal position of text to print to the right of H in inches - -1 causes the text to be centered in PCL mode. -1 is not compatible with NWP.
TEXT$	Text string to be printed starting at TV TH
CPI	
FONT$	

Comments:

FNPRINTFORM$*40

```
FNPRINTFORM$*40(FILNUM,FORMFILE,SHORTNAME$)
```

Description
Extracts a form,page,macro or font from a library file and places it into an existing open display file for printing

Functions used

Variables:

FILNUM	Number of existing open print file to receive form. The file should be opened with EOL=NONE.
FORMFILE	Number of existing open file containing the form to be printed.
SHORTNAME$	Eight character name for the storied form. This is the key-name within the FORMFILE

Comments:
The function reads through the records of the FORMFILE until a match for the SHORTNAME is found. That record along with subsequent records containing the same SHORTNAME are added to the open print file. Records are transferred in 32000 bit chunks so the transfer is quite rapid.

FNMACROTEMP - Makes a PCL macro temporary

```
FNMACROTEMP(filnim,macnr$;delete)
```

Description
The function takes a macro variable previously generated by FNPRINTFORM and does an SREP to change the PCL macro call into a statement that makes the macro a temporary macro in the printer. See also FNMACROPERM to perform the same function, except make the macro permanent.

A macro included in a print file is active in the printer only while the print file is being processed. In order for the macro to be available to other print files it must be assigned a status of TEMPORARY or PERMANENT. A subsequent print file can then use the macro by executing the macro call without the necessity of reloading the macro to the printer, a considerable time saver.

Functions used

Variables:

FILNUM	Number of open print file through which the temporary assignment will be transferred to the printer
MACNR$	macro call previously generated by FNPRINTFORM. The macro call is generally in the form of chr$(27)&"&200y3X" where 200 is the number of the macro that will be affected.
DELETE	An optional parameter which. if non-zero, will cause the macro to be deleted from printer memory.

Comment:
When printing multiple forms in separate print files it becomes very inefficient to continue loading large macros. By loading an overlay macro at the beginning of the first print file and the calling the macro with each subsequent print file a considerable amount of time can be saved.

FNMACROPERM - Makes a PCL macro temporary

```
FNMACROPERM(filnim,macnr$;delete)
```

Description
The function takes a macro variable previously generated by FNPRINTFORM and does an SREP to change the PCL macro call into a statement that makes the macro a permanent macro in the printer. See also FNMACROTEMP to perform the same function, except make the macro temporary.

A macro included in a print file is active in the printer only while the print file is being processed. In order for the macro to be available to other print files it must be assigned a status of TEMPORARY or PERMANENT. A subsequent print file can then use the macro by executing the macro call without the necessity of reloading the macro to the printer, a considerable time saver.

Functions used

Variables:

FILNUM	Number of open print file through which the temporary assignment will be transferred to the printer
MACNR$	macro call previously generated by FNPRINTFORM. The macro call is generally in the form of chr$(27)&"&200y3X" where 200 is the number of the macro that will be affected.
DELETE	An optional parameter which. if non-zero, will cause the macro to be deleted from printer memory.

FNSIGNBOX - Prints a signature or small graphic in PCL

```
FNSIGNBOX(FILNUM,V,H,SIGFIL,SHORT$,&PASS$)
```

Description
Extract a small graphic such as a signature from a library file and place it at a specified location on a document

Functions used

Variables:

FILNUM	Number of open print file to which the signature should be added
V	Vertical position in inches of the upper left corner of the graphic to print
H	Horizontal position in inches of the left hand edge of the graphic to print
SIGFIL	The number of the file containing the signature graphic
SHORT$	The eight character name of the signature to be used. If this case sensitive name is not found in the file no signature is printed
&PASS$	Password - case sensitive. Must match the password saved for the signature or no signature will be printed. The password is passed back to the application so that on a check run or similar application the operator will not have to enter the password for each check.

Comments:
The signature file is built using a separate utility program names SIGPRN.br. The signature is taken from the print file created by printing a Word document containing just the signature to an HP6L laser printer driver in print to file mode. The utility program print a facsimile of the signature as part of the import process. The facsimile is overlain with reference lines showing where the upper left corner of the print graphic appears.

The signature is limited to one 32000 bit record. Consequently large or complex signatures or graphics may not be compatible and may need to be made smaller or less compiles in order to work with this particular program.

RTF Printing

FNRTFSTART - opens a source file to produce an RTF file using RTFLIB.dll

Opens a file ready to receive data for creating an RTF file using LIBRTF.dll

```
FNRTFSTART(HEADER$*100,FOOTER$*100,TITLE$*500,MAT HEADER$;CELLNO)
```

Functions used:

Variables:

HEADER$	Text to be displayed as a header on each page of the report
FOOTER$	Text to be displayed as a footer on each page of the report. To include a page number include "[PAGE]" as a part of the line.
TITLE$	Text to be displayed at the top of the first page only as a report title
MAT HEADERS$	The matrix including the bar delimited text that should appear in the header bar at the top of each column
CELLNO	An optional cell number for the SPC file if a header that repeats automatically on each page is to be used. If omitted the headers will appear on the first page only formatted exactly the same as the rest of the RTF table that is being created.

Comments:

==========

FNRTFEND - turns a source file built with FNRTFSTART into a finished document

Converts

```
FNRTFEND$*100(RTFNO,RTFNAME$*100,RTFSPEC$*100;WORD)
```

Functions used:

Variables:

RTFNO	The file number of the display file that was opened when FNRTFSTART was called
RTFNAME$	The name and path of the source file to be created when RTFNO is processed by FNRTF to RTF.
RTFSPEC$	The name of the RF specification file that contains style formats to be used in creating the RTF file
WORD	A flag to indicate whether WORD should be called at the end of the creation process (True) or the RTF file should not be viewed at the end of the process (false)

Comments:

A sample specification file looks like the following

```
LET LMARGIN=.75
```

```
LET RMARGIN=1.0
LET TMARGIN=.50
LET BMARGIN=.50
LET ORIENTATION$="PORTRAIT"
LET PAPER$="LETTER"
LET CHECKLIST=0
LET LEFTTEXT$=""
LET NUME=0
MAT TYPES$(12)
LET TYPES$(1)="H"
LET TYPES$(2)="F"
LET TYPES$(3)="D"
LET TYPES$(4)="S"
LET TYPES$(5)="T"
LET TYPES$(6)="A"
LET TYPES$(7)="B"
LET TYPES$(8)="C"
LET TYPES$(9)="E"
LET TYPES$(10)="G"
LET TYPES$(11)="N"
LET TYPES$(12)="I"
MAT STYLES$(12)
LET STYLES$(1)="li0|ri0|fARIAL|fs14|cfBlue|tc3.25|Header"
LET STYLES$(2)="li0|ri0|fARIAL|fs8|cfBlack|tc3.25|Footer"
LET
     STYLES$(3)="li0.5|QJ|fPALATINO|ri0|fs12|tl0.5|tl1.0|tl1.5|td5.4||Dat
     a"
LET STYLES$(4)="li0.5|QC|sa1|ri0|B|fs19|fARIAL|tl0.5||tc3.25|Title Page"
LET STYLES$(5)="li0.5|QC|fARIAL|sa1|ri0|B|fs18|tl0.5||tc3.25|Heading 1"
LET STYLES$(6)="li0.25|ri0|fARIAL|B|fs17|tl0.5||tr5.4|Heading 2"
LET STYLES$(7)="li0.25|ri0|fARIAL|B|fs15|tl0.5||tr5.4|Heading 3"
LET STYLES$(8)="li0.25|ri0|fARIAL|B|fs13|tl0.5||td5.4|Heading 4"
LET STYLES$(9)="fi-
     0.5|td0.75|li1.0|ri0|fPALATINO|fs12|tl0.5|tl1.0|td6.0|Detail steps"
LET STYLES$(10)="fi-
     0.4|li1.0|ri0|ft61|fs10|fCOURIER|tl0.5|tc4.0|td5.4|Program lines"
LET
     STYLES$(11)="li0.5|ri0|B|fPALATINO|fs12|cfDKBLUE|tl0.5|tl1.0|tl1.5|t
     d5.4||New Items"
LET STYLES$(12)="fi-1.25|li2.0|ri0|fPALATINO|fs12|tl2.0|Options"

MAT CELLS$(10)
rem LET CELLS$(1)="li0.5|tg0.125|c1|btrlb1|vt|hl|c1.5|btrlb1|vt|hc|"
rem LET
     CELLS$(2)="li0.5|tg0.125|fPALATINO|fs10|c3|btrlb1|vt|hl|c3|btrlb1|vt
     |hl|"
rem LET
     CELLS$(3)="li0.5|tg0.125|fPALATINO|fs10|c2|btrlb1|vt|hl|c2|btrlb1|vt
     |hl|c2|btrlb1|vt|hl|"

rem ODD numbers are headers even numbers are the following table

LET CELLS$(1)="li0.0|tg0.100|fPALATINO|fs10|trh|"
LET CELLS$(1)=CELLS$(1)&"c3.0|brtlrb1|vt|hc|sh15|"
LET CELLS$(1)=CELLS$(1)&"c3.0|brtlrb1|vt|hc|sh15|"

LET CELLS$(2)="li0.0|tg0.100|fPALATINO|fs10|"
LET CELLS$(2)=CELLS$(2)&"c3.0|brtrlb1|vt|hl|"
LET CELLS$(2)=CELLS$(2)&"c3.0|brtrb1|vt|hl|"

LET CELLS$(3)="li0.0|tg0.100|fPALATINO|fs10|trh|"
```

```
LET CELLS$(3)=CELLS$(3)&"c2.0|brtrlb1|vt|hc|sh15|"
LET CELLS$(3)=CELLS$(3)&"c2.0|brtrlb1|vt|hc|sh15|"
LET CELLS$(3)=CELLS$(3)&"c2.0|brtrlb1|vt|hc|sh15|"

LET CELLS$(4)="li0.0|tg0.100|fPALATINO|fs10|"
LET CELLS$(4)=CELLS$(4)&"c2.0|btrlb1|vt|hl|"
LET CELLS$(4)=CELLS$(4)&"c2.0|btrlb1|vt|hl|"
LET CELLS$(4)=CELLS$(4)&"c2.0|btrlb1|vt|hl|"

LET CELLS$(5)="li0.5|tg0.100|fARIAL|fs10|"
LET CELLS$(5)=CELLS$(5)&"c3.0|btrlb1|fCOURIER|vt|hl|"
LET CELLS$(5)=CELLS$(5)&"c0.5|fs8|btb1|vt|hc|"
LET CELLS$(5)=CELLS$(5)&"c0.5|btlb1|fPALATINO|vt|hc|"
LET CELLS$(5)=CELLS$(5)&"c1|fs10|btrlb1|fARIAL|vt|hr|"
LET CELLS$(5)=CELLS$(5)&"c1|brtrlb1|vt|hr|"
LET CELLS$(5)=CELLS$(5)&"c1|brtrlb1|vt|hr|"
LET CELLS$(5)=CELLS$(5)&"c1|brtrlb1|vt|hr|"
LET CELLS$(5)=CELLS$(5)&"c1|brtrlb1|vt|hr|"
LET CELLS$(5)=CELLS$(5)&"c1|brtrlb1|vt|hr|"

LET CELLS$(7)="li0.0|tg0.100|fPALATINO|fs10|trh|"
LET CELLS$(7)=cells$(7)&"c0.5|brtrlb1|vt|hc|sh15|"
LET CELLS$(7)=cells$(7)&"c0.5|brtrlb1|vt|hc|sh15|"
LET CELLS$(7)=cells$(7)&"c0.5|brtrlb1|vt|hc|sh15|"
LET CELLS$(7)=cells$(7)&"c0.5|brtrlb1|vt|hc|sh15|"

LET CELLS$(8)="li0.0|tg0.100|fPALATINO|fs10|"
LET CELLS$(8)=cells$(8)&"c0.5|brtrlb1|vt|hc|"
LET CELLS$(8)=cells$(8)&"c0.5|brtrlb1|vt|hc|"
LET CELLS$(8)=cells$(8)&"c0.5|brtrlb1|vt|hc|"
LET CELLS$(8)=cells$(8)&"c0.5|brtrlb1|vt|hc|"
```

==========

FNREFERENCE - Prints a page reference on bottom right corner in PCL

```
FNREFERENCE(PTYPE$,REFERENCE$;PFILE,LGL)
```

Description
Prints a page reference in PCL in the lower right corner of a printed page

Functions used

Variables:

PTYPE$	Must start with "HP" in order for the reference code to be printed
REFERENCE$	Reference code to be printed in lower right corner of page
PFILE	The number of the currently open print file where the reference should be inserted
LGL	If True print for a legal sized page otherwise print for letter sized

Comments:

FNPRINT_FILE - Prints a text file on Grey bar Paper

```
FNPRINT_FILE(FILE_NAME$*100;INDENT)
```

Description
Prints an ASCII file formatted at 100 character lines with a ruler at the top of the page

Functions used

Variables:

FILE_NAME$	Name of display file to print on greybar paper
INDENT	Number of spaces that each line of text should be indented from the left margin

Comments:

FNPRINTERS - Creates a printed list of printers and a printers.sys file

```
FNPRINTERS(;DRIVE_LOC$)
```

Description:
Not to be confused with the PRINTER.SYS PCL/NWP substitution parameters

Functions used

Variables:

Comments:

Date and Time

Date formatting

FNCCYYMMDD_TO_DAYS - converts CYMD to DAYS

Converts CYMD to DAYS

```
FNCCYYMMDD_TO_DAYS(&DAT)
```

Functions used

Variables:

DAT date to be converted in DAYS format

Comments:

FNDATEFWD

```
FNDATEFWD(DATEIN;CENTURY)- Converts YYMMDD to MMDDYY
```

Description
Converts YYMMDD to MMDDYY with option of including century

Functions used None

Variables:

DATEIN date in YYMMDD format

Comments:

If century is true then the date is output in MDCY format

FNDATEREV - converts MMDDYY to YYMMDD

```
FNDATEREV(DATEIN;CENTURY) ! Convert MMDDYY to YYMMDD with optional
     addition of century MMDDCCYY to YYMMDD if century >0
```

Description
Converts MMDDYY to YYMMDD with option of including century

Functions used

Variables:

DATEIN	date in MMDDYY format
CENTURY	1 if century is to be included CCYY
	0 if no century, only year YY

Comments:

FNDATE$ - Creates a formatted date from DAYS input

Creates a formatted date from a DAYS input. Example January 5,2003 Beginning with 4.17 this can be done directly by BR.

```
FNDATE$(DAYSIN)
```

Functions used

Variables:

DAYSIN	date to be converted in DAYS format

Comments:

FNDAYS_TO_MMDDCCYY - Converts DAYS to MDCY

Converts DAYS to MDCY

```
FNDAYS_TO_MMDDCCYY(&DAT)
```

Functions used

Variables:

DAT	date to be converted in DAYS format

Comments:

FNDAYS_TO_MMDDYY - Converts DAYS to MMDDYY

Converts DAYS to MDY

```
FNDAYS_TO_MMDDYY(&DAT)
```

Functions used

Variables:

DAT	date to be converted in DAYS format

Comments:

FNMDY2YMD - converts YYMMDD to MMDDYY with century option

Converts YYMMDD to MMDDYY with option of including century

```
FNMDY2YMD(DATEIN;CENTURY)
```

Functions used

Variables:

DAYSIN	date to be converted in DAYS format
CENTURY	1 if century is to be included 0 if no century

Comments:

FNMMDDCCYY_TO_DAYS - converts MDCY to DAYS

Converts MDCY to DAYS

```
FNMMDDCCYY_TO_DAYS(&DAT)
```

Functions used

Variables:

DAT date to be converted in DAYS format

Comments:

FNMMDDYY_TO_DAYS - Converts MDY to DAYS

Converts MDY to days

```
FNMMDDYY_TO_DAYS(&DAT)
```

Functions used

Variables:

DAT	date to be converted in DAYS format

Comments:

FNTIMMILREG - 12 hour time from 24 hour time

Returns regular 12 hour time from 24 hour military time

```
FNTIMMILREG(MILTIM,&HOUR,&MINUTES,&AMPM$) !:
```

Functions used

Variables:

MILTIM	time in military 24 hour format
HOUR	the hour in 12 hour time to be returned
MINUTES	minutes to be returned
AMPM$	designation of AM or PM to be returned

Comments:

FNYMD2MDY - converts MMDDYY to YYMMDD with century option

Converts MMDDYY to YYMMDD with option of including century

```
FNYMD2MDY(DATEIN;CENTURY)
```

Functions used

Variables:

DAYSIN	date to be converted in DAYS format
CENTURY	1 if century is to be included, 0 if no century

Comments:

FNYYMMDD_TO_DAYS - convert YMD to DAYS

Converts YMD to DAYS

```
FNYYMMDD_TO_DAYS(&DAT)
```

Functions used

Variables:
DAT date to be converted in DAYS format

Comments:

Relative and special dates

FNBUSINESSDAY - returns the next business day after or including a specified date

increases the given date until it is not a weekend or legal holiday

```
FNBUSINESSDAY(XDATE)
```

Functions used:

Variables:
XDATE Date in days that is the starting point for calculations

Comments:
Useful in determining settlement dates for federal and state tax payments.
The holidays calculated are federal legal banking holidays only, no state or local holidays are included.

==========

FNDAYOFYEAR - ordinal number of days from beginning of calendar year

Calculates what day any days date is within the calendar year that it is located

```
FNDAYOFYEAR(D)
```

Functions used:

Variables:
D Date in days to be calculated

Comments:

==========

FNNEXTMONTH - similar date in the following month

Returns a similar date for the following month based on number of days before month end

```
FNNEXTMONTH(INDATE)
```

Functions used

Variables:

INDATE	date to be converted in DAYS format

Comments:
If the date given is the 29th day of a 31 day month and the following month contains 30 days the returned value will be the 28th day of the following month.

FNPRIOR BUSINESSDAY - returns the first business day prior to a given date including the given date

Decreases the given date until it is not a weekend or legal holiday

```
FNPRIOBUSINESSDAY(XDATE)
```

Functions used:

Variables:

XDATE	Seed date in days

Comments:
Will calculate the available date for a banking transaction prior to or including the seed date. Useful in determining the settlement date for payroll direct deposit dating and payroll account funding in advance of a payroll.

==========

FNWEEKDAY$ - Returns the day of the week

Returns the day of the week from a DAYS input

```
FNWEEKDAY$(WEEKDAY) !:
```

Functions used

Variables:

WEEKDAY	date to be converted in DAYS format

Comments:

FNWEEKOFMONTH - number of time a specified day of week has occur ed in the month specified

Returns the week number within a month of a specified date assuming that the first time that that day of the week occurred in the month was the first week of the month.

```
FNWEEKOFMONTH(D)
```

Functions used:

Variables:

D	Date for which the calculation is being done

Comments:
Useful in determining what deduction in a payroll system should be activated if the deductions only occur on certain weeks of the month.

==========

FNWEEKOFYEAR - number of times a specified day of week has occurred in a year up to a specified date

Returns the number of the week of the year for a specified date, assuming that the specified day of the week first occurred in the first week of the year.

```
FNWEEKOFYEAR(D)
```

Functions used:

Variables:

D	Date to be processed in days

Comments:

==========

Array Functions

Sorting arrays

FNSORTARRAY - sort an array with header and footer

Sorts an array either ascending or descending and optionally excludes elements at the top and bottom to allow headers and footers to remain in place - sort is based on positions within the array, not the start of the string

```
FNSORTARRAY(MAT L$,START,LENGTH;DESENDING,HEADER,FOOTER)!:
```

Functions used None

Variables:

Mat L$	matrix to be sorted
START	Starting position for the character string on which to sort
LENGTH	length of the character sub string on which to sort
DESCENDING	if true sorts descending order else sorts ascending
HEADER	number of rows at the top of the matrix to omit from the sort
FOOTER	number of rows at the bottom of the matrix to omit from the sort

Comments:

Sort an array on any character sub-set allowing for header rows at the top and footer/total rows at the bottom.

FNSRTARY - sort an array with header and footer based on itself

Similar to FNSORTARRAY but uses the entire string to sort rather than a sub string

```
FNSRTARY(MAT L$;MAT M$,DESENDING,HEADER,FOOTER)
```

Functions used None

Variables:

Mat L$	matrix to be sorted
DESCENDING	if true sorts descending order else sorts ascending
HEADER	number of rows at the top of the matrix to omit from the sort
FOOTER	number of rows at the bottom of the matrix to omit from the sort

Comments:

FNSRTNARY -Sort a numeric array based on another array

Similar to FNSRTARY but for a numeric matrix

```
FNSRTNARY(MAT L;MAT M$,DESENDING,HEADER,FOOTER) !:
```

Functions used None

Variables:

Mat L$	matrix to be sorted
DESCENDING	if true sorts descending order else sorts ascending
HEADER	number of rows at the top of the matrix to omit from the sort
FOOTER	number of rows at the bottom of the matrix to omit from the sort

Comments:

Array arithmetic

FNCOLSUM - sums the elements of an array for a specified column

Provides the sum of a single column of a multi column array.

```
FNCOLSUM(MAT L,C)
```

Functions used:

Variables:

MAT L	Matrix containing multiple columns
C	Column number to be summed

Comments:

==========

FNROWSUM - sums the elements of an array for a specified row

Returns the sum of a row of a multi row and multi column array

```
FNROWSUM(MAT L,R)
```

Functions used:

Variables:

Mat L	Numeric array containing the row to be totaled
R	Row number to be totaled

Comments:

==========

Searching arrays

FNCHRMAT$ - convert a numeric array to character

Convert a numeric matrix into a formatted character matrix

```
FNCHRMAT$(CHRMAT$,NUMMAT,FORMAT$;BLANKS)
```

Functions used None

Variables:

Mat CHRMAT$	matrix that will be output
Mat NUMMAT	numeric matrix being converted
FORMAT$	format used to convert each line to a string
BLANKS	if true replaces a zero value with blanks

Comments:

FNLISTSRCH - searches a character array based on a search string

```
FNLISTSRCH(MAT L$,SRCHSTR$,MAT SELECT;STRT)
```

Description
Performs a search on a matrix and modifies the matrix select with elements in the searched matrix that match SRCHSTR$

Functions used

Variables:

Mat L$	The array to be searched. The search is case insensitive and will match any matching combination regardless of position within each element.
SRCHSTR$	The string that is being matched to each element, case insensitive
MAT SELECT	A numeric array that holds the row numbers of matching elements. Any newly found elements are added to the array.
STRT	Optional positioning number. matches will only occur if the match is AFTER this position in the row string

Comments:
Used in lists and grids following FNLISTSPEC to allow for a search of the arrays used in a list or grid and a positioning of the cursor on elements matching the criteria

FNLISTSRCHN - searches a numeric array based on a search string

```
FNLISTSRCHN(MAT L,SRCHSTR$,MAT SELECT;STRT,SMASK$)
```

Description
Same as FNLISTSRCH except for a numeric matrix

Functions used

Variables:

Mat L	The array to be searched. Each element is turned into a string before being searched. The search is case insensitive and will match any matching combination regardless of position within each element.
SRCHSTR$	The string that is being matched to each element, case insensitive

MAT SELECT	A numeric array that holds the row numbers of matching elements. Any newly found elements are added to the array.
STRT	Optional positioning number. matches will only occur if the match is AFTER this position in the row string

Comments:
Used in lists and grids following FNLISTSPEC to allow for a search of the arrays used in a list or grid and a positioning of the cursor on elements matching the criteria

FNSELECTION - selection process using two arrays

```
FNSELECTION(SELECTION,MAT SEL$,MAT SEL;MANY)
```

Description
Maintains two matrices, one SEL is true if an item is selected. The other SEL$ contains the selection sequence number if MANY is greater than one or the word SELECTED if MANY equals one. If many=0 only one item is allowed as a selection.

Functions used None

Variables:

SELECTION	the element number selected or deselected MAT SEL$ selection number or word MAT SEL true if element is selected MANY 0 for a single selection 1 for any or all and a number for a limited number of elements

Comments:

FNSRCHCRIT$*50 - search criteria for a list box

```
FNSRCHCRIT$*50(SR$,SC$,LROWS,LCOLS,PARENT;MESSAGE$)
```

Description
Opens a window within a listbox window and asks for a search string

Functions used

Variables:

Comments:

Other

FNDELROW - removes a row from an array and redimensions the array

Removes a row form an array and redimensions the array to be one row shorter

```
FNDELROW$(MAT DEL,DELROW)
```

Functions used:

Variables:

MAT DEL	The numeric array that needs to be updated
DELROW	The row number to delete

Comments:

==========

FNDELROW$ - removes a row from an array and redimensions the array

Removes a row form an array and redimensions the array to be one row shorter

```
FNDELROW$(MAT DEL$,DELROW)
```

Functions used:

Variables:

MAT DEL$	The character array that needs to be updated
DELROW	The row number to delete

Comments:

==========

FNPARMAT - split an array into sub-arrays

Parses an array into a multi-dimensional array based on splitting at a predefined character

```
FNPARMAT(MAT M$,SUB$;NOREF)
```

Functions used None

Variables:

MAT M$	matrix to be parsed
MAT M$	matrix to be parsed
SUB$	character that will be treated as a boundary or field separator
	NOREF if true prevents a single line matrix from being reformatted to a one dimensional matrix

Comments:

Miscellaneous Functions

Email

FNEMAIL - creates an email file for email monitor

Creates an email file for EMAILMONITOR

```
FNEMAIL(SENDDIR$*80,MAILFROM$*50,SUBJECT$*100,MAT MAILTO$,MAT
     MESSAGE$;MAT ATTACH$,SMAILQ$*80)
```

Functions used

Variables:

SENDDIR$ directory where message will be created
MAILFROM$ sender's email address
SUBJECT$ subject line of email
MAT MAILTO$ email addresses of recipients
MAT MESSAGE$ email text in the form of a matrix
MAT ATTACH$ matrix containing full path and name of any attachments

Comments:

EMAILMONITOR is available through David Blankenship

FNEMAILFILE - inserts a text file into an email for email monitor

Inserts a text file into an email for EMAILBLASTER

```
FNEMAILFILE(SENDDIR$*80,MAILFROM$*50,SUBJECT$*100,MAT
     MAILTO$,TEXTFILE$*100;MAT ATTACH$,SMAILQ$*80)
```

Functions used

Variables:

SENDDIR$ directory where message will be created
MAILFROM$ sender's email address
SUBJECT$ subject line of email
MAT MAILTO$ email addresses of recipients
MAT MESSAGE$ email text in the form of a matrix
TEXTFILE$ name of file containing email message

MAT ATTACH$matrix containing full path and name of any attachments

Comments:

Formatting

FNLEADZERO$ - obsolete replace with CNVRT$("PIC(###)",x)

Converts a number to a string and fills the leading positions with "0"'s.

```
FNLEADZERO$(NUMBER,LENGTH)
```

Functions used:

Variables:

NUMBER	The number to be converted
LENGTH	The length of the resulting field

Comments:
Easier done with the CNVRT$("PIC(#####)",number) function.

FNCHECKAMOUNT$ - returns English words for a dollar amount

Converts a number into a string of English words formatted with the words Dollars and Cents. Optionally allows the returned string to be left padded with tilde symbols.

```
FNCHECKAMOUNT$(AMOUNT;LENGTH,OPT)
```

Functions used:
The routine uses a local function to convert each three number (hundreds, thousands, millions) into words for the final result.

Variables:

AMOUNT	The number to be converted. This will be truncated to two decimal places. Maximum number is 999,999,999.99. A zero or negative number will return the word VOID.
LENGTH	An optional prameter. If used and greater than 10 the result will be left padded with tilde symbols to the size specified. If the result is "V O I D" the word VOID will be centered in the padded tildes.
OPT	An option parameter to determne whether the words DOLLARS and CENTS are included in the output string. 0 will include these words, 1 will transform the cents to a fraction and include it prior to the final word dollars. 2 will transform the cents to a fraction and append it to the output string, but with no "Dollars" included so that the string can be added to a preprinted check.

Comments:
Designed to be used as check protection verbiage on computer printed checks. Can als be used as a screen response description.

Progress

FNPROG - displays a progress bar for a process

Displays a vertical progress bar that changes color form green to yellow to red as the process approaches 100%

```
FNPROG(PROW,PCOL,PCUR,PTOT)
```

Functions used:

Variables:

PROW	Upper left row corner of display
PCOL	Upper left column corner of display
PCUR	Current record number
PTOT	Total record numbers when project is complete

Comments:
If reading a file the file needs to be restored after obtaining the last record number

FNPROGRESS - displays a progress bar for a process

Similar to FNPROG

```
FNPROGRESS(&PCT_WINDEV,PCT_TOTAL,PCT_DONE;SR$,CAPTION$*55)
```

Functions used:

Variables:

PCT_WINDEV	
PCT_TOTAL	Total number of transactions to complete....
PCT_DONE	Number of transactions completed
SR$	Starting row for display
CAPTION$	Optional window caption

Comments:

==========

Other

FNCLKBUF - clears the keyboard buffer of extra key strokes

Clears the keyboard buffer

```
FNCLKBUF
```

Functions used:

Variables:

None

Comments:

==========

FNCURDRV$ - returns the current drive and directory

Returns the current drive and directory

```
FNCURDRV$
```

Functions used:

Variables:
NONE

Comments:

==========

FNMSEXE$ - return the installed location of a Microsoft compliant program installation

Uses David Blankenship's BRREGISTER2.exe to query the registry for the installed location of registered software

```
FNMSEXE$(L$)
```

Functions used:

Variables:
L$ executable name as registered in the registry.

Comments:
Will find the location of WINWORD.EXE, EXCEL.EXE or any other executable that i properly registered

==========

FNX$ - returns X if true BLANK if false

Returns an "X" if L is true or " " if L is false.

```
FNX$(L)
```

Functions used:

Variables:
None

Comments:

==========

FNSNAP Obsolete functions

Window and screen processing

FNMGCLR - clears a message form the fnpick message line

```
FNMGCLR ! Clear message and reset error processing flags
```

Description
Clears the message line form the old FNSNAP message line setup

Functions used

Variables:

Comments:

FNSAVPART - legacy function to save a portion of the screen - obsolete in 4.17

```
FNSAVPART(SR$,SC1$,ER$,EC$,CLEARIT) !:
```

Description
Saves a portion of the screen - this is a legacy from character days

Functions used

Variables:

Comments:

FNRELPART - legacy function to clear a portion of the screen obsolete in 4.17

```
FNRELPART(SCRREF,RESTSCR) !:
```

Description

Functions used

Variables:

Comments:

FNWIN - legacy function to open a window in numeric order

```
FNWIN(SR$,SC1$,ER$,EC$,WINTITL$*80,BORDTYP$*32,WINCOL$,WINNUM,DIMLST) !:
```

Description
Opens a window and assigns an incremental number to the window.

Functions used

Variables:

Comments:

FNCLSWIN - legacy function to close a window opened by FNWIN

```
FNCLSWIN(CLRWIN) !:
```

Description
Closes the last window opened by FNWIN and sets the window number to zero

Functions used

Variables:

Comments:

FNPM - legacy message box on the main window

```
FNPM(TXT$*78;CENTER) !:
```

Description
Prints a message on the message line and optionally center the text.

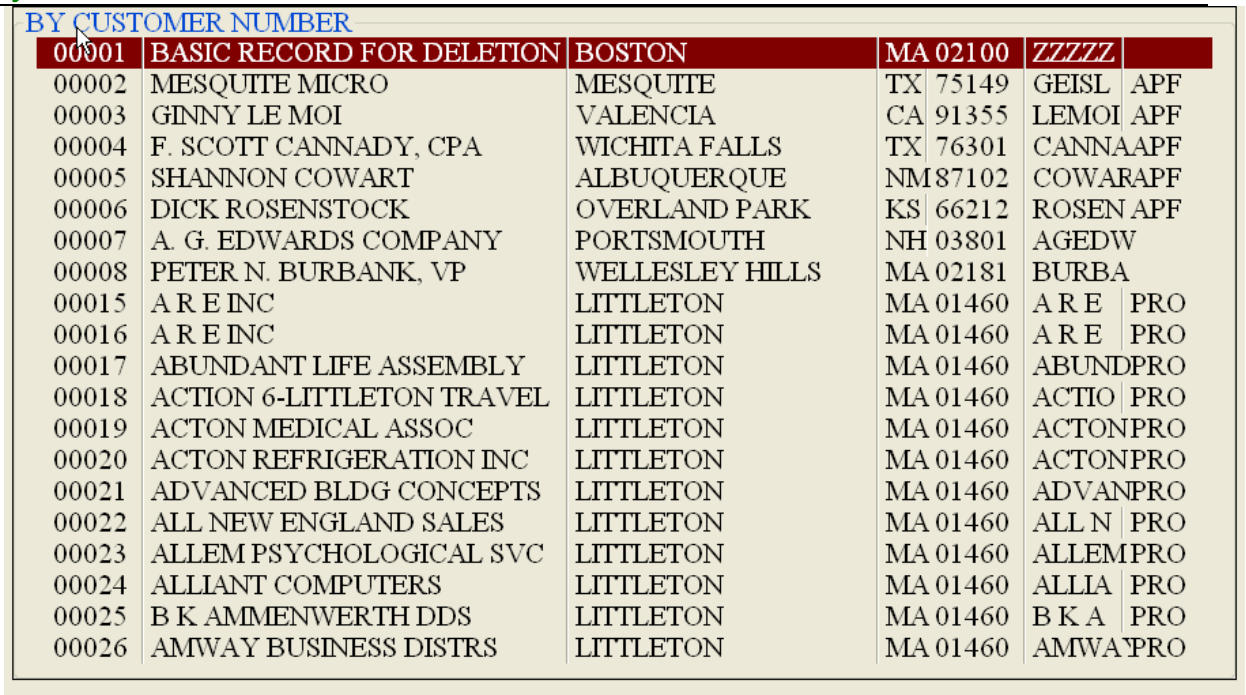
Functions used

Variables:
The message line must have been previously defined in the library. See FNINIT

Comments:

Point and pick

FNKEYSEL - direct file look up function requires a fixed position font


BY CUSTOMER NUMBER

00001	BASIC RECORD FOR DELETION	BOSTON	MA 02100	ZZZZZ	
00002	MESQUITE MICRO	MESQUITE	TX 75149	GEISL	APF
00003	GINNY LE MOI	VALENCIA	CA 91355	LEMOI	APF
00004	F. SCOTT CANNADY, CPA	WICHITA FALLS	TX 76301	CANNA	APF
00005	SHANNON COWART	ALBUQUERQUE	NM 87102	COWAR	APF
00006	DICK ROSENSTOCK	OVERLAND PARK	KS 66212	ROSEN	APF
00007	A. G. EDWARDS COMPANY	PORTSMOUTH	NH 03801	AGEDW	
00008	PETER N. BURBANK, VP	WELLESLEY HILLS	MA 02181	BURBA	
00015	A R E INC	LITTLETON	MA 01460	A R E	PRO
00016	A R E INC	LITTLETON	MA 01460	A R E	PRO
00017	ABUNDANT LIFE ASSEMBLY	LITTLETON	MA 01460	ABUND	PRO
00018	ACTION 6-LITTLETON TRAVEL	LITTLETON	MA 01460	ACTIO	PRO
00019	ACTON MEDICAL ASSOC	LITTLETON	MA 01460	ACTON	PRO
00020	ACTON REFRIGERATION INC	LITTLETON	MA 01460	ACTON	PRO
00021	ADVANCED BLDG CONCEPTS	LITTLETON	MA 01460	ADVAN	PRO
00022	ALL NEW ENGLAND SALES	LITTLETON	MA 01460	ALL N	PRO
00023	ALLEM PSYCHOLOGICAL SVC	LITTLETON	MA 01460	ALLEM	PRO
00024	ALLIANT COMPUTERS	LITTLETON	MA 01460	ALLIA	PRO
00025	B K AMMENWERTH DDS	LITTLETON	MA 01460	B K A	PRO
00026	AMWAY BUSINESS DISTRS	LITTLETON	MA 01460	AMWA	PRO

Figure: SNAP0012.ptf

```
FNKEYSEL(SROW$,SCOL$,PP,&KEY$,FILENBR,FORM$*100,BT$*80,BTP$,MAXL,HK$*40,H
     LPFIL$*80,HLPELE) !:
```

Description

A legacy direct point and shoot listing of records in a file. Character based and not GUI looking.

Functions used

Variables:

Comments:

FNPICK_EX - A point and shoot legacy using a single matrix requires a fixed position font

1040AK	Arkansas Indivi	375.00	375.00	375.00	375.00	375.00
1040AL	Alabama Individ	500.00	500.00	500.00	500.00	500.00
1040AZ	Arizona Individ	500.00	500.00	500.00	500.00	500.00
1040CA	California Indi	625.00	625.00	625.00	625.00	625.00
1040CA_PLR	CAL 1040 PLR RE	10.00	10.00	10.00	10.00	10.00
1040CO	Colorado Indivi	375.00	375.00	375.00	375.00	375.00
1040CONV_E	ELECTRONIC CONV	2.50	2.50	2.50	2.50	2.50
1040CT	Connecticut Ind	625.00	0.00	0.00	0.00	0.00
1040CT_PLR	CT 1040 PLR RET	10.00	10.00	10.00	10.00	10.00
1040DC	Dist. of Columb	375.00	375.00	375.00	375.00	375.00
1040DE	Deleware Indivi	375.00	375.00	375.00	375.00	375.00
1040FL	Florida Individ	250.00	250.00	250.00	250.00	250.00
1040FL_PLR	FL 1040 per ret	10.00	10.00	10.00	10.00	10.00
1040GA	Georgia Individ	500.00	500.00	500.00	500.00	500.00
1040HW	Hawaii Individu	500.00	500.00	500.00	500.00	500.00
1040ID	Idaho Individua	375.00	375.00	375.00	375.00	375.00
1040IL	Illinois Individ	500.00	500.00	500.00	500.00	500.00
1040IL_PLR	Illinois per re	10.00	0.00	0.00	0.00	0.00
1040IN	Indiana Individ	500.00	0.00	0.00	0.00	0.00
1040IW	Iowa Individual	500.00	0.00	0.00	0.00	0.00

Figure: SNAP0011.ptf

```
FNPICK_EX(PICK_OPS,SROW$,SCOL$,PP,MAT
     L$,WINTITLE$*80,BORDTYPE$,MAXL,HK$*40,HLPFIL$*80,PTYP,HLPELE,MANY,RP
     TFCOL,AUTOSEL,MAT SEL_TYPES$,&MUSTRSET,SEARCHON,SSTR$*40;MAT
     SEL,&SCPT,&XKEY,MAT PICKWIN)
```

Description

Functions used

Variables:

Comments:

FNPICK - A point and shoot legacy using a single matrix requires a fixed position font

```
FNPICK(PICK_OPS,SROW$,SCOL$,PP,MAT
     L$,WINTITLE$*80,BORDTYPE$,MAXL,HK$*40,HLPFIL$*80,PTYP,HLPELE,MANY,RP
     TFCOL,AUTOSEL,MAT SEL_TYPES$,&MUSTRSET,SEARCHON,SSTR$*40;MAT
     SEL,&SCPT,&XKEY)
```

Description

Functions used

Variables:

Comments:

FNKEYSEL_EX - a legacy direct file access point and shoot requires a fixed sized font

```
FNKEYSEL_EX(SROW$,SCOL$,PER_WIN,&KEY$,FILENBR,KFORM$*100,NBRFIELDS,RETKEY
     FIELD,ACTKEYFIELD,CANCELKEY,WINTITLE$*80,BORDTYPE$,TOTLENGTH,HK$*40,
     HLPFIL$*40,HLPELE;KEY_CHECK,KEY_CHECK_FIELD$*30) !:
```

Description

Functions used

Variables:

Comments:

FNPOPUP - legacy code for a pop-up choice box

```
FNPOPUP(MAT MOPT$,MAT HOTKEY$,SROW$,SCOL$,MENUTITLE$*80,MENUBORDER$,MAT
     HM$,HMROW,HK$*40,HLPFIL$*60,OPLEN,POPNUM;POPRESET) !:
```

Description
Created a pop up list box with options. This is an old character based function that has been converted to use the new gui.

Functions used

Variables:

Comments:

Supporting functions

Data transfer between program and library

FNPUTPICKWIN - transfers pick window number to FNSNAP form program

```
FNPUTPICKWIN(MAT PPICKWIN)
```

Description
Transfers the PICKWIN matrix from the calling program to FNSNAP

Functions used None

Variables:
MAT PPICKWIN matrix to transfer

Comments:

FNGETPICKWIN - retrieve pick window matrix from FNSNAP

```
FNGETPICKWIN(MAT PPICKWIN)
```

Description
Transfers the PICKWIN matrix from FNSNAP to the calling program

Functions used None

Variables:
MAT PPICKWIN matrix to transfer

Comments:

FNPICKWIN - gets the current pick window from fnsnap library

```
FNPICKWIN(WIN,VALWIN) !:
```

Description

Functions used

Variables:

Comments:

FNWINDEV - returns the number of the currentlyactive FNSNAP window using the old system

If FNWIN is being used to open windows then FNWINDEV returns the number of the most recently opened, and still open, window. This proactive is becoming obsolete with GUI applications.

```
FNWINDEV
```

Functions used:

Variables:
None

Comments:

==========

FNLEADZERO$

```
FNLEADZERO$(NUMBER,LENGTH)- converts and zero fills a number
```

Turns a number into a zero filled string, similar to CNVRT$(PIC(#####)",number)

Functions used

Variables:

Comments:

FNGETK$ -

```
FNGETK$(X) !:
```

Get X key strokes and return the uppercase unhexed value if it is a letter

Functions used

Variables:

Comments:
Used by some FNSNAP functins to create seach strings

FNSETALL - set all elements of an array

```
FNSETALL(SFLG) ! Set ALL elements of MAT SEL & MAT L$ 1/0
```

Description

Functions used

Variables:

Comments:

FNSETSEL - set all elements of an array

```
FNSETSEL(ELE,SETFLG) ! Set L$(ELE) for setflg 1-ON,0-OFF
```

Description

Functions used

Variables:

Comments:

FNPRTPICKBAR - a function to position a colored pick bar in FNPICK

```
FNPRTPICKBAR(COLOR$,PICK_ELE) ! print the fnpick light bar for COLOR$,
     element PICK_ELE
```

Description

Functions used

Variables:

Comments:

FNZLPAD$ - pads a number with zeros and converts to string

```
FNZLPAD$(NUMBER,LENGTH,DECIMALS)
```

Description

Functions used

Variables:

Comments:

FNINIT - initializes the variable required by the original FNSNAP functions

Converts

```
FN()
```

Functions used:

Variables:

FILENAME$
DIRNAME$

Comments:

==========

FNNOKEY - chekc to see if CMDKEY or FKEY were pressed

Check to see if a CMDKEY or FKEY was pressed and produce an error for field C

```
FNNOKEY(c)
```

Functions used:

Variables:

C Field number to produce an error for

Comments:
Used by some older FNSNAP Utilities

==========