MikePack 1.65

4th Dimension® External Package

User Documentation

Introduction

Before delving head first into the gory guts of the world of syntax and code examples, I wanted to give you some background on the MikePack package, and why it exists.

MikePack started out as a group of a few loosely related 4D routines that I seemed to use in almost all of my projects. I decided to set up a library of 4D routines that I could use from project to project. That worked, but some of the routines were kind of slow. Pascal came to the rescue. I re-wrote all of the 4D library routines in Pascal, and had a set of 6 or 8 externals.

As I took on new projects, and new needs arose, this little grouping turned into over 90 routines in 5 separate packages!

I decided that these routines were certainly not "rocket science", they were simply faster, or easier ways to do things I was already doing in 4D. That's why they're shareware, not commercial.

I've broken the manual down into separate sections. There's a table of contents, index, and one section of detail per package. The routines are listed within their respective sections.

For each routine, there is a syntax explanation, a table describing parameter and return value usage, and a paragraph or two of how to use the command. There's also a code example for most of the commands.

I hope that these routines save you time and effort in your 4D adventures, and that you can find new and inventive ways to use them. The demo database simulates several pieces of general business routines, but hopefully you can devise even more creative methods of using MikePack!

Thanks for your interest and time!

Mike Jimenez

General Information

All of the routines that deal with arrays in any way, will require you to pass the **NAME** of the array, and **NOT THE ARRAY** itself. This is VERY important.

The routine MP DISTINCT, requires that the arrays be sorted before use in the external.

Except where noted, ALL arrays managed by these packages are to be TEXT arrays. Two notable exceptions are contained in the MP S2N_Array and MP N2S_Array routines. Please consult the parameter tables for more details.

The Array Routines

MP Array2File

Err := MP Array2File ("myArray" ; FileName ; Creator ; Type)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
myArray	STRING	NO	**Name** of array to work with
FileName	STRING	NO	Full pathname of file to be created
Creator	STRING	NO	4 digit file creator descriptor
Type	STRING	NO	4 digit file type descriptor
RETURNS VALUE	**TYPE**		**DESCRIPTION**
YES	INTEGER		Mac OS File system error codes from trying to create the file. 0 means all went well.

The data in the TEXT array will be written to a file. A system error will be returned into Err, 0 means successful creation of the file.

A full pathname can be used as the file name. Creator can be any valid 4 character id, ie. "MSWD" for Microsoft Word, or "XCEL" for Microsoft Excel. Type should also be a 4 character descriptor.
Use "TEXT".
If you feel extremely experimental, you could use something else...

MP Array2Text

myText := MP Array2Text ("myArray" ; Delimiter)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
myArray	STRING	NO	**Name** of array to work with
Delimeter	STRING	NO	Character to act as separator between lines of the text field
RETURNS VALUE	**TYPE**		**DESCRIPTION**
YES	TEXT		The text created by the merging of the lines of the array.

The contents of the array will be placed into the text variable. Pass the name of the array, not the array itself. The char(Delimeter) parameter will separate the elements of the array as they're placed into the text variable.

MP FILE2TEXT

MP FILE2TEXT (FileName ; Err ; "TextArray")

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
FileName	STRING	NO	Full pathname of file to be read into memory
Err	STRING	YES	Error number from OS
TextArray	STRING	NO	**Name** of array to receive contents of file
RETURNS VALUE	**TYPE**		**DESCRIPTION**
NO	N/A		N/A

Pass the filename to be read, the error variable (text), and the text array to place the file into. I used an array because files can easily be larger than 32k (the limit of text variables). The data is moved into the array in 32k chunks.

The errors returned are Mac OS errors.

MP MERGEARRAYS

MP MERGEARRAYS ("aList" ; Delimeter ; "BigArray")

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
aList	STRING	NO	**Name** of the array which contains the names of the arrays to be merged
Delimeter	STRING	NO	Delimeter to separate the columns of the created array
BigArray	STRING	NO	**Name** of the array to receive the merged results
RETURNS VALUE	**TYPE**		**DESCRIPTION**
NO	N/A		N/A

Pass the name of an array that contains the names of the arrays to be merged. (what?) See the demo. Also pass the character you want to use as a delimiter (if any) between columns (for no delim, pass ""). Lastly, pass the array to receive the merged results.

MP TEXT2ARRAY

MP TEXT2ARRAY (vText ; "yArray , DelimNum)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
vText	TEXT	NO	Text variable to be broken down into an array
myArray	ARRAY	NO	Array to receive the results of the procedure
DelimNum	INTEGER	NO	ASCII number of the delimiter character that breaks lines of text
RETURNS VALUE	**TYPE**		**DESCRIPTION**
NO	N/A		N/A

Pass the char(DelimNum) delimited text variable, and a populated text array will be created.

NOTE : In versions previous to 1.65, the *NAME* of the array was passed, now the array itself is passed. This is *VERY IMPORTANT*. The program will crash if this is not done correctly.

MP FILLARRAY

MP FILLARRAY ("ResultArray" ; "SourceArray" ; Where)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
ResultArray	STRING	NO	**Name** of the array to receive the elements from the source array
SourceArray	STRING	NO	**Name** of the array that the elements are to be moved from
Where	INTEGER	NO	Element number in the result array to start inserting elements from SourceArray
RETURNS VALUE	**TYPE**		**DESCRIPTION**
NO	N/A		N/A

This routine moves the data from "SourceArray" into "ResultArray" starting at element Where. All data is inserted into the array, and previous elements are adjusted "downward".

If Where = Size of Array(ResultArray) + 1 then the SourceArray will be appended to ResultArray.

MP S2N_Array

MP S2N_Array ("Source" ; "Result")

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
Source	STRING	NO	**Name** of the array that contains the Text elements
Result	STRING	NO	**Name** of the array that the Text elements are to moved and converted into
RETURNS VALUE	**TYPE**		**DESCRIPTION**
NO	N/A		N/A

This routine moves the data from "Source" into "Result", converting the string data in "source" to numeric data in "result".

MP N2S_Array

MP N2S_Array ("Source" ; "Result")

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
Source	STRING	NO	**Name** of the array that contains the numeric elements
Result	STRING	NO	**Name** of the array that the numeric elements are to moved and converted into
RETURNS VALUE	**TYPE**		**DESCRIPTION**
NO	N/A		N/A

This routine moves the data from "Source" into "Result", converting the numeric data in "Source" to string data in "Result".

MP POPULATE

MP POPULATE ("Source" ; Value)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
Source	STRING	NO	**Name** of the array to be populated
Value	STRING	NO	Value to place in every element of Source
RETURNS VALUE	**TYPE**		**DESCRIPTION**
NO	N/A		N/A

This routine places "Value" into all elements of the "Source" array.

MP DISTINCT

MP DISTINCT ("Source" ; "Result")

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
Source	STRING	NO	**Name** of the array that contains the sorted elements
Result	STRING	NO	**Name** of the array that will receive the distinct elements from Source
RETURNS VALUE	**TYPE**		**DESCRIPTION**
NO	N/A		N/A

This routine fills array "Result" with the distinct values of array "Source". Very similar to the "Distinct Values" command in v3.0.x, but works on arrays.

MP SearchArray

NumElems:= MP SearchArray ("ToFind" ; aSource ; Start ; aResults ; aIndexNums ; "Operator")

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
ToFind	STRING	NO	The text to be located in the array - wild cards are allowed
Source	ARRAY	NO	Text array that will be searched
Start	INTEGER	NO	Element in Source array to begin searching for ToFind
Results	ARRAY	NO	Text array to receive the values from the source that match the search criteria
ElementNums	ARRAY	NO	LongInt array to receive the element numbers from the source where matches were found
ComparisonOP	STRING	NO	Operator : "=", "#", "<", ">", "<=", ">="
RETURNS VALUE	**TYPE**		**DESCRIPTION**
YES	INTEGER		Returns the number of elements found in the source array that match the search criteria

This routine returns the number of occurrences of "ToFind" in aSource. It also populates aResults with the values found in aSource, and populates aIndexNums with the element numbers the values were found at. "Start" is the first element to be searched. "Operator" is =, #, <, >, <=, >=.

NOTE
Pass the actual arrays, NOT the names to this routine

MP ARRAYSELECT

MP ARRAYSELECT ("aValues" ; FileNumber ; FieldNumber)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
Values	STRING	NO	**Name** of the array that contains the values to be found in the file
FileNumber	INTEGER	NO	File number of file to be searched.
FieldNumber	INTEGER	NO	Field number of field to be used in searching
RETURNS VALUE	**TYPE**		**DESCRIPTION**
NO	N/A		N/A

This routine creates a selection in the given file where the given field equals a value in the aValues array.

Can be used in conjunction with MP SearchArray to create a selection based on several elements of a given array.

MP APPLY2ARRAY

MP APPLY2ARRAY ("aSource" ; "FuncProc" ; ChangeElemsYN)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
Source	STRING	NO	**Name** of the array that contains the Text elements
FuncProc	STRING	NO	Function or procedure to execute
ChangeElemsYN	INTEGER	NO	Flag for deciding to change the elements with the result of the function - 0 = No, 1 = Yes
RETURNS VALUE	**TYPE**		**DESCRIPTION**
NO	N/A		N/A

This routine executes the "FuncProc" once for every element in the "aSource" array. If the "FuncProc" is a function and returns a value, then ChangeElemsYN is looked at, and if it = 1 then the result of the function is placed in the current element of the array, and if it's 0, nothing is changed in the array.

To substitute the current element of the array into the function/procedure, place the name of the array, and a ^ in place of the element number :

MP APPLY2ARRY("aNames";"Uppercase(aNames{^})";1)

or

MP APPLY2ARRAY("aNums";"aNums{^}:=aNums{^-1}+1";0)

The ^ is simply replaced in the function by the current element number, so you can do math with it as in the above example.

You can also issue procedure calls :

```
MP APPLY2ARRAY("aValues";"SEND PACKET(Doc;aValues{^})";0)
```

The Drag & Drop Routines

MP DragBlock

myRegion := MP DragBlock ("Left";"Top";"Right";"Bottom";Width;Height)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
Left	STRING	NO	**Name** of the integer array containing the left coordinates of the drop-off rects
Top	STRING	NO	**Name** of the integer array containing the top coordinates of the drop-off rects
Right	INTEGER	NO	**Name** of the integer array containing the right coordinates of the drop-off rects
Bottom	STRING	NO	**Name** of the integer array containing the bottom coordinates of the drop-off rects
Width	INTEGER	NO	Width, in pixels, of the rect to be dragged
Height	INTEGER	NO	Height, in pixels, of the rect to be dragged
RETURNS VALUE	**TYPE**		**DESCRIPTION**
YES	INTEGER		Returns the drop-off rect number that the rect was dropped in

Pass the names of the arrays containing the coordinates of the valid "drop off" regions for the block being dragged. All coordinates are local to the window they appear in. Also pass the width and height of the block to be dragged.
The region into which the block was dropped will be returned.

Errors : -1 = Not dropped into a valid region
-2 = Not all the arrays are the same size
-3 = More than 10 regions.

MP DragText

myRegion := MP DragText (myStr ; "Left" ; "Top" ; "Right" ; "Bottom")

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
StringToDrag	STRING	NO	Text to use to create gray rect for dragging
Left	STRING	NO	**Name** of the integer array containing the left coordinates of the drop-off rects
Top	STRING	NO	**Name** of the integer array containing the top coordinates of the drop-off rects
Right	INTEGER	NO	**Name** of the integer array containing the right coordinates of the drop-off rects
Bottom	STRING	NO	**Name** of the integer array containing the bottom coordinates of the drop-off rects
RETURNS VALUE	**TYPE**		**DESCRIPTION**
YES	INTEGER		Returns the drop-off rect number that the rect was dropped in

Pass the string to be dragged, and the names of the arrays containing the coordinates of the valid "drop off" regions for the block being dragged. The region into which the block was dropped will be returned.

Errors : -1 = Not dropped into a valid region
-2 = Not all the arrays are the same size
-3 = More than 10 regions.

MP MultiDrag

myRegion := MP MultiDrag("Left";"Top";"Right";"Bottom";"Windows";W;H)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
Left	STRING	NO	**Name** of the integer array containing the left coordinates of the drop-off rects
Top	STRING	NO	**Name** of the integer array containing the top coordinates of the drop-off rects
Right	INTEGER	NO	**Name** of the integer array containing the right coordinates of the drop-off rects
Bottom	STRING	NO	**Name** of the integer array containing the bottom coordinates of the drop-off rects
Windows	STRING	NO	**Name** of the text array containing the names of the windows the drop-off rects are in
Width	INTEGER	NO	Width, in pixels, of the rect to be dragged
Height	INTEGER	NO	Height, in pixels, of the rect to be dragged
RETURNS VALUE	**TYPE**		**DESCRIPTION**
YES	INTEGER		Returns the drop-off rect number that the rect was dropped in

Pass the names of the arrays containing the coordinates of the valid "drop off" regions for the block being dragged. Also pass the names of the windows containing the regions. All region coordinates are local to the windows they appear in. Also pass the width and height of the block to be dragged.
The region into which the block was dropped will be returned.

***Note
This routine will hilight the valid dropoff rects as they are dragged over.

Errors : 0 = Dropped on the desktop
-1 = Not dropped into a valid region
-2 = Not all the arrays are the same size
-3 = More than 10 regions.

MP DragItem

myRegion := MP DragItem (myStr;"Left";"Top";"Right";"Bottom";"Windows")

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
StringToDrag	STRING	NO	Text to drag around the screen
Left	STRING	NO	**Name** of the integer array containing the left coordinates of the drop-off rects
Top	STRING	NO	**Name** of the integer array containing the top coordinates of the drop-off rects
Right	INTEGER	NO	**Name** of the integer array containing the right coordinates of the drop-off rects
Bottom	STRING	NO	**Name** of the integer array containing the bottom coordinates of the drop-off rects
Windows	STRING	NO	**Name** of the text array containing the names of the windows the drop-off rects are in
RETURNS VALUE	**TYPE**		**DESCRIPTION**
YES	INTEGER		Returns the drop-off rect number that the rect was dropped in

Pass the string to be dragged, and the names of the arrays containing the coordinates of the valid "drop off" regions for the block being dragged. The region into which the block was dropped will be returned.

***Note
This routine will hilight the valid dropoff rects as they are dragged over.

Errors : -1 = Not dropped into a valid region
-2 = Not all the arrays are the same size
-3 = More than 10 regions.

The Gestalt Routine

MP Gestalt

myResult := MP Gestalt (Selector ; Result)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
Selector	STRING	NO	4 Character selector for Gestalt
Result	LONGINT	YES	Numeric representation of answer to question
RETURNS VALUE	**TYPE**		**DESCRIPTION**
YES	LONGINT		Error code from Gestalt

The selector is a valid 4 character string, and the result is a long integer result.

See Inside Macintosh VI, chapter 3, page 46 for more info on selectors and responses. All of the MP k... commands are the valid selectors for MP Gestalt.

Example : Err := MP Gestalt ("mach";mType)
mType : 11 = IIci, 18 = IIsi, etc.

Below are all of the selector constants in MikePack 1.6

MP kAddrMode	MP kFileSys	MP kMemMgr	MP kResMgr
MP kAliasMgr	MP kFileTrans	MP kMiscAttr	MP kROMSize
MP kAppleEvts	MP kFolders	MP kNotifyMgr	MP kROMVersion
MP kAppleTalk	MP kFonts	MP kNuBusAttr	MP kScrMgr
MP kAUX	MP kFPU	MP kNumScripts	MP kSerialHW
MP kCommRsrc	MP kGestaltMgr	MP kOSAttr	MP kSoundMgr
MP kCommTools	MP kHardware	MP kOSTraps	MP kStdFileAttr
MP kConnect	MP kHelpMgr	MP kParity	MP kStdNPB
MP kCPU	MP kMP keyBoard	MP kPhysRAM	MP kSysVersion
MP kDAM	MP kLowMem	MP kPopups	MP kTerminalMgr
MP kDialogs	MP kLPageSize	MP kPowerMgr	MP kTextEdit
MP kEasyAccess	MP kLRAMSize	MP kPPC	MP kTimeMgr
MP kEditionMgr	MP kMachIcon	MP kQDFeatures	MP kTrapTable
MP kExtTools	MP kMachineType	MP kQDVersion	MP kVMem

The PopupMenu Routines

MP PopupMenu

myItem := MP PopupMenu ("ItemList" ; DefaultItem ; H ; V)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
Items	STRING	NO	**Name** of array containing the elements of a popup menu
Default	STRING	NO	Item which is to be the default item
Horizontal	INTEGER	NO	Horizontal location, in pixels, for the upper left of the menu to appear
Vertical	INTEGER	NO	Vertical location, in pixels, for the upper left of the menu to appear
RETURNS VALUE	**TYPE**		**DESCRIPTION**
YES	INTEGER		Which item was chosen from the menu

This routine creates a popup menu containing the items in the array "ItemList", starts the user at item DefaultItem, and draws the menu at coordinates H,V.

It returns the chosen menu item, or 0 for none.

MP PopupPlus

myItem := MP PopupPlus ("ItemList" ; DefaultItem ; H ; V ; CheckYN)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
Items	STRING	NO	**Name** of array containing the elements of a popup menu
Default	STRING	NO	Item which is to be the default item
Horizontal	INTEGER	NO	Horizontal location, in pixels, for the upper left of the menu to appear
Vertical	INTEGER	NO	Vertical location, in pixels, for the upper left of the menu to appear
CheckYN	INTEGER	NO	1 = Check the default item, 0 = Don't check the default item
RETURNS VALUE	**TYPE**		**DESCRIPTION**
YES	INTEGER		Which item was chosen from the menu

This routine creates a popup menu containing the items in the array "ItemList", starts the user at item DefaultItem, and draws the menu at coordinates H,V. If CheckYesNo = 0, then the default item is not checked. If it = 1, it is.

Note

In MP PopupPlus to create a sub menu, a menu item should be in this form :

[SubMenuTitle]SubMenuArrayName

ex. [Colors]aColors
where aColors is a text array with the names of colors in it.

For MP PopupPlus : if myItem > 99 then :
SubMenu Number = INT(myItem/100)
SubMenu Item = myItem - (INT(myItem/100)*100)

The String Routines

MP JustifyText

myString := MP JustifyText (myString ; myLen ; character ; myType)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
String	TEXT	NO	String to be placed into a field of characters
Length	INTEGER	NO	Length of the field to be created
Character	STRING	NO	Character to use to fill in the field, around the text being justified
JustificationType	STRING	NO	"L" = Left justify "C" = Center "R" = Right justify
RETURNS VALUE	**TYPE**		**DESCRIPTION**
YES	STRING		The justified text

Pass the original string, the length of the desired resulting string, the character to use as a pad, and the type of justification :

"L" = Left "C" = Centered "R" = Right

MP PadText

myText := MP PadText (myText ; myLen ; Character)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
String	STRING	NO	String to add padding to
Length	INTEGER	NO	Total size of padded text
Character	STRING	NO	Character to pad (Add to the right) the text with
RETURNS VALUE	**TYPE**		**DESCRIPTION**
YES	STRING		The padded string

This will pad the passed text on the right side to the desired length using the passed character.

(Same as justifying text as being "Left" justified).

MP TrimText

myText := MP TrimText (myText)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
String	STRING	NO	String that is to have leading and trailing spaces removed from it
RETURNS VALUE	**TYPE**		**DESCRIPTION**
YES	STRING		Original string with no leading or trailing spaces

Pass the text to be trimmed, and all leading and trailing spaces will be summarily exterminated! (removed).

MP TrimLeft

myText := MP TrimLeft (myText)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
String	STRING	NO	String that is to have leading spaces removed from it
RETURNS VALUE	**TYPE**		**DESCRIPTION**
YES	STRING		Original string with no leading spaces

Pass the text to be trimmed, and all leading spaces will be summarily exterminated! (removed).

MP TrimRight

myText := MP TrimRight (myText)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
String	STRING	NO	String that is to have trailing spaces removed from it
RETURNS VALUE	**TYPE**		**DESCRIPTION**
YES	STRING		Original string with no trailing spaces

Pass the text to be trimmed, and all trailing spaces will be summarily exterminated! (removed).

MP SCROLLTEXT

MP SCROLLTEXT ("aLines";"aSizes";"aTechs";"aFonts";"aStyles";"aColors";vWidth;vHeight)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
Lines	STRING	NO	**Name** of the array that contains the lines of text to be scrolled
Sizes	STRING	NO	**Name** of the Integer array that contains the sizes of the fonts for each line
Techniques	STRING	NO	**Name** of the Integer array that contains the display techniques, 0 or 1, for each line
Fonts	STRING	NO	**Name** of the array that contains the font names to use for each line
Styles	STRING	NO	**Name** of the array that contains the style to use for each line
Colors	STRING	NO	**Name** of the array that contains the color of the text for each line
Width	INTEGER	NO	Width of the display window for the scrolling text
Height	INTEGER	NO	Height of the display window for the scrolling text
RETURNS VALUE	**TYPE**		**DESCRIPTION**
NO	N/A		N/A

Will open a window and scroll the lines of text through the window until a click. The size, font, etc. is set for each line. The "aTechs" array contains the "techniques" used to display each line : Either 0 = Place in the center of the line, or 1 = slide the two halves of the line together.

Colors : BLACK, WHITE, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW
Styles : Plain, Bold, Italic, Underline

MP DRAWTEXT

MP DRAWTEXT("Howdy";H;V;Ticks;"FontName";Size;"StyleNumber")

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
String	STRING	NO	Text to be drawn on the screen
Horizontal	INTEGER	NO	Horizontal location to draw the text at
Vertical	INTEGER	NO	Vertical location to draw the text at
Ticks	INTEGER	NO	Number of ticks to leave the text on the screen before erasing it - 0 = Don't erase
Font	STRING	NO	Font Name for text
Size	INTEGER	NO	Size of text
Style	STRING	NO	Style of text
RETURNS VALUE	**TYPE**		**DESCRIPTION**
NO	N/A		N/A

In the front most window, will draw the passed text at the passed size, font, and style number. It will stay on the screen for "Ticks" ticks (1/60) of a second. If you pass 0 ticks, it will not erase.

Styles : Plain, Bold, Italic, Underline

MP SCROLLRECT

MP SCROLLRECT (left ; top ; right ; bottom ; dx ; dy)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
Left	INTEGER	NO	Left side of rect to be scrolled
Top	INTEGER	NO	Top side of rect to be scrolled
Right	INTEGER	NO	Right side of rect to be scrolled
Bottom	INTEGER	NO	Bottom side of rect to be scrolled
Hor. Distance	INTEGER	NO	Number of pixels to scroll the area horizontally
Vert. Distance	INTEGER	NO	Number of pixels to scroll the area vertically
RETURNS VALUE	**TYPE**		**DESCRIPTION**
NO	N/A		N/A

Pass the coordinates of the rect to scroll, and the number of pixels to scroll in the x and y directions. -x = left, -y = up.

Scrolls **EVERYTHING** in the rect passed to it.

MP FRAMERECT

MP FRAMERECT (left ; top ; right ; bottom ; w ; h)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
Left	INTEGER	NO	Left side of rect to be drawn
Top	INTEGER	NO	Top side of rect to be drawn
Right	INTEGER	NO	Right side of rect to be drawn
Bottom	INTEGER	NO	Bottom side of rect to be drawn
Width	INTEGER	NO	Width of line being drawn
Height	INTEGER	NO	Height of line being drawn
RETURNS VALUE	**TYPE**		**DESCRIPTION**
NO	N/A		N/A

Pass the coordinates of the rect to draw, and the number of pixels to use as the pen size.

MP ERASERECT

MP SCROLLRECT (left ; top ; right ; bottom ; w ; h)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
Left	INTEGER	NO	Left side of rect to be erased
Top	INTEGER	NO	Top side of rect to be erased
Right	INTEGER	NO	Right side of rect to be erased
Bottom	INTEGER	NO	Bottom side of rect to be erased
Width	INTEGER	NO	Width of line being erased
Height	INTEGER	NO	Height of line being erased
RETURNS VALUE	**TYPE**		**DESCRIPTION**
NO	N/A		N/A

Pass the coordinates of the rect to erase, and the number of pixels to use as the pen size.

The Clipboard Routines

This is a new section for MikePack v1.6

MP Array2Clip

This routine has been moved from the Array Pack as of version 1.6

```
myResult := MP Array2Clip ( "myArray" )
```

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
Source	STRING	NO	**Name** of the array that contains the Text elements to be copied to the clipboard
RETURNS VALUE	**TYPE**		**DESCRIPTION**
YES	INTEGER		Clipboard error code

Will place the contents of the text array onto the clipboard, returning any errors along the way.

MP Pict2Clip

```
myResult := MP Pict2Clip ( "PictVar" )
```

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
PictureVar	STRING	NO	**Name** of the picture variable that is to be copied to the clipboard
RETURNS VALUE	**TYPE**		**DESCRIPTION**
YES	INTEGER		Clipboard error code

Will place the contents of the picture variable onto the clipboard, returning any errors along the way.

The Window Routines

This is a new section for MikePack v1.6

MP WINDOWLOC

MP WINDOWLOC (vLeft ; vTop ; vRight ; vBottom)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
Left	INTEGER	YES	Receives the left side of the frontmost window, in global coordinates
Top	INTEGER	YES	Receives the top side of the frontmost window, in global coordinates
Right	INTEGER	YES	Receives the right side of the frontmost window, in global coordinates
Bottom	INTEGER	YES	Receives the bottom side of the frontmost window, in global coordinates
RETURNS VALUE	**TYPE**		**DESCRIPTION**
NO	N/A		N/A

Returns the global coordinates of the front most window (in pixels).

MP WINDOWSIZE

MP WINDOWSIZE (vHeight ; vWidth)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
Height	INTEGER	YES	Receives the height of the frontmost window, in global coordinates
Width	INTEGER	YES	Receives the width of the frontmost window, in global coordinates
RETURNS VALUE	**TYPE**		**DESCRIPTION**
NO	N/A		N/A

Returns the height and width of the front most window (in pixels).

MP SIZEWINDOW

MP SIZEWINDOW (vHeight ; vWidth)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
Height	INTEGER	NO	Sets the height of the frontmost window, in pixels
Width	INTEGER	NO	Sets the width of the frontmost window, in pixels
RETURNS VALUE	**TYPE**		**DESCRIPTION**
NO	N/A		N/A

Sets the height and width of the front most window (in pixels).

MP MOVEWINDOW

MP MOVEWINDOW (H ; V)

PARAMETER	TYPE	RECEIVES DATA	DESCRIPTION
Horizontal	INTEGER	NO	Sets the left side of the frontmost window, in pixels
Vertical	INTEGER	NO	Sets the top side of the frontmost window, in pixels
RETURNS VALUE	**TYPE**		**DESCRIPTION**
NO	N/A		N/A

Sets the top left corner of the front most window to H,V (in pixels).

Index