Functions.hyper

COLLABORATORS

	TITLE : Functions.hyper	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		December 6, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Functions.hyper

1.1 Function Reference (Tue Nov 3 16:46:29 1992)

Contents:

Functions:

```
   pubscreen()      (return name of public screen)
   qual()           (get qualifier from last pressed key)
   realloc()        (reallocate memory)
   rfrate()         (return refresh rate)
   rfcmd()          (return pointer to refresh command string)
   stsize()         (get size of structure)
   taglist()        (get current tag list number)
   toppc()          (give program counter at top of debug log window)

Various:
   Back to main contents
```

1.2 Function Reference : Introduction

```
This reference file contains all functions. Commands are not in this file.
Note that you can use these functions from ARexx. Simply use the same
format as for normal commands :

   A function call in PowerVisor : res=if(a>b,1,2)
   How you must do this in ARexx : 'if a>b 1 2'
                                   res=result
```

1.3 Function Reference : alloc()

```
   <memory> = ALLOC( ('n',<size>) | ('c',<size>) | ('s',<string>) )


Use this function for easy memory allocation.
Use the  free()  function to free this memory, use the  getsize()  function
to get the size of this memory and use the  realloc()  function to
reallocate the memory.

The memory allocated with this function is guaranteed to contain
only 0 (except if the first argument is equal to 's').

Note that memory allocated with this command is automatically freed
when PowerVisor quits.

When the blocksize is smaller than 65533 bytes the result from this
function is a pointer after a word containing the size. This pointer is
thus word alligned but not longword alligned.
When the blocksize is bigger the result is a pointer after a longword
containing the size. This pointer is longword alligned.
Using word or longword allignment you can determine the size of a memory
block. An easier way to do this is to use the  getsize()  function.


Example :


To allocate 1000 bytes :
```

```
< a=alloc(n,1000) <enter>
```

To allocate 11 bytes and copy the string 'PowerVisor' to this memory use :

```
< a=alloc(s,'PowerVisor') <enter>
```

To allocate 1000 bytes in chip ram use :

```
< a=alloc(c,1000) <enter>
```

Related commands: cleanup showalloc

Related functions: free() realloc() getsize() isalloc()

1.4 Function Reference : apeek()

```
<address> = APEEK( <structure pointer>,<struct def pointer>,
            <field name> )
```

Returns the address of <structure>.<field name>. <struct def pointer> must be a pointer to a previously loaded structure definition. <structure pointer> is the pointer to the structure itself. <field name> must be defined in the structure definition.

'apeek' uses autodefault to the 'stru' list for the second argument.

Related commands: addstruct interprete struct

Related functions: peek() stsize()

Related lists: stru

Related tutor chapters: Looking at things

1.5 Function Reference : arexxport()

```
<pointer to name> = AREXXPORT( )
```

Returns the pointer to the name of the ARexx port for this instance of PowerVisor.
This function returns a string if used from ARexx.

The name of the PowerVisor ARexx port is :

```
REXX_POWERVISOR
```

```
or if you are running a slave instance (the first instance of PowerVisor
is the master. All following instances running at the same time with the
master are slaves) :

   REXX_POWERVISOR.<num>


Related commands:  rx

Related functions:  pubscreen()

Related tutor chapters:  Scripts
```

1.6 Function Reference : base()

```
   <pointer> = BASE( )


This function returns the pointer to the first element in the current list.


Example :

Go to the task list :
< task <enter>

< list <enter>
> Task node name      : Node     Pri   StackU   StackS Stat Command         Acc
> ------------------------------------------------------------------------------
> ConClip Process     : 07E60410 00       242     4000 Wait sys:c/ConCl(02) -
> RexxMaster          : 07E6AA48 04       162     2048 Wait             (00) -
> « IPrefs »          : 07E59568 00       862     3500 Wait             PROC -
> ClickToFront        : 07E75210 15       398     4096 Wait             PROC -
> CpuBlit             : 07E7BA18 00       266     2048 Wait             PROC -
> ...

< d base() <enter>
> 07E60410 , 132514832


Related commands:  list
```

1.7 Function Reference : botpc()

```
   <programcounter> = BOTPC( )


This function returns the program counter visible at the bottom of the
'Debug' logical window.
You can set this program counter using the  dstart  or  dscroll  commands.
```

Related commands: debug dwin dscroll dstart

Related functions: toppc() isbreak() getdebug()

Related lists: dbug

Related tutor chapters: Debugging

1.8 Function Reference : checksum()

```
<checksum> = CHECKSUM( <address>,<bytes> )
```

Returns a checksum for the given memory range. This function transforms
<address> and <bytes> to longword alligned values.

1.9 Function Reference : cols()

```
<cols> = COLS( <logwin> )
```

This function returns the maximum number of columns available on the
logical window.

'cols' uses autodefault to the 'lwin' list for the first argument.

Example :

```
< disp cols(main) <enter>
> 00000056 , 86
```

Related commands: colrow fit

Related functions: lines() getcols() getrows() getlwin() getx()
gety()

Related lists: lwin

Related tutor chapters: Screens and windows

1.10 Function Reference : curlist()

```
<pointer to string> = CURLIST( )
```

Return a pointer to the name of the current list.
This function returns a string if used from ARexx.

```
Note for experienced users only !
   The pointer to the string is actually a pointer into the infoblock
   for the current list. curlist()-24 points to the start of the infoblock.
   See  The wizard corner  for more information about infoblocks (in the
   'info base').
   Note that if you want to use this feature in ARexx you have to call
   this function using  void  or  assign . If you call it directly you
   will get a string and not a pointer in ARexx


Related tutor chapters:  Getting Started   The wizard corner
```

1.11 Function Reference : eval()

```
   <result> = EVAL( <string pointer> )


Evaluate the string at <string pointer> and return the result.
'eval' returns 0 if <string pointer> is 0.


Example :

< disp eval("1+2") <enter>
> 00000003 , 3

< scan <enter>
????< 4+4 <enter>
< disp eval(input) <enter>
> 00000008 , 8


Related commands:  disp   void

Related functions:  if()

Related tutor chapters:  Expressions
```

1.12 Function Reference : free()

```
   FREE( <pointer> )


Free a block previously allocated with  alloc() .
Note that all memory allocated with 'alloc' is automatically freed when
PowerVisor quits or when you use  cleanup .


Related commands:  cleanup   showalloc

Related functions:  alloc()   realloc()   getsize()   isalloc()
```

1.13 Function Reference : getactive()

```
<logwin> = GETACTIVE( )
```

This function returns a pointer to the active logical window. This is the logical window where you can scroll with the keyboard. You can change the active logical window with the active command or with the <tab> key.

Related commands: active on current

Related functions: getlwin()

Related lists: lwin

Related tutor chapters: Screens and windows

1.14 Function Reference : getchar()

```
<char> = GETCHAR( )
```

Get the character at the current screenposition in the current logical window.

Example :

```
< {locate 10,10;a=getchar()} <enter>
```

Related commands: locate home cls current

Related functions: getx() gety() getcols() getrows() cols() lines() getlwin() getline()

Related lists: lwin

Related tutor chapters: Screens and windows

1.15 Function Reference : getcol()

```
<columns> = GETCOL( <logical window> )
```

Get the real number of columns (or -1) for the logical window. Note that this is not always equal to the visible number of columns. If <columns> is not equal to -1 it is the total number of columns. If <columns> is equal to -1 it means that the logical window is autoscalable in horizontal direction. To get the real number of columns (after scaling) you can use

```
the  cols()  function.

'getcol' uses autodefault to the 'lwin' list for the first argument.


Related commands:  colrow   fit

Related functions:  getrow()   cols()   lines()   getlwin()   getx()
                    gety()

Related lists:  lwin

Related tutor chapters:  Screens and windows
```

1.16 Function Reference : getdebug()

```
   <debug node> = GETDEBUG( )


Return the current debug node. You can set the current debug node with
 duse  and  with .


Related commands:  duse   with   debug

Related functions:  isbreak()   toppc()   botpc()

Related lists:  dbug

Related tutor chapters:  Debugging
```

1.17 Function Reference : geterror()

```
   <error> = GETERROR( <expression string> )


Evaluate an expression and return the error. If there was no error, 0
is returned.


Example :

< disp geterror("7") <enter>
> 00000000 , 0

0 since 7 is a valid expression.

< disp geterror("7+") <enter>
> 00000010 , 16

returns 16 since there is a missing operand.
```

The following errors are defined (Note ! Some of these are obsolete and will never appear) (Note ! you can change the errorstrings since this file is called PowerVisor-errors (you can find this file in the program directory where the PowerVisor executable is or in the S: directory)) :

-6 Stack overflow !!!
-5 A program has crashed !!!
-4 A stack overflow was getting close !
-3 Quiting PowerVisor !
-2 PowerVisor has crashed !
-1 Break...
0 No error
1 Not enough memory !
2 Syntax Error !
3 This is not a device created with OpenDev !
4 Bad list element type !
5 Variable is a constant !
6 Only <B>yte, <W>ord or <L>ong !
7 Odd address error !
8 Could not Lock!
9 Bracket '(' expected !
10 To many arguments for library function !
11 Missing operand !
12 Lock is not a subdirectory !
13 Error while opening device !
14 Unknown list element !
15 Not implemented yet !
16 Unknown mode argument !
17 Unknown AddFunc argument !
18 This is not a process !
19 Node is not a task or process !
20 This task is not frozen!
21 This task is already frozen!
22 Node type is wrong !
23 Addressed element not found !
24 Window is not sizeable !
25 This is no supported library function !
26 No help available for this subject !
27 Error while opening file !
28 Error while reading file !
29 Not a resident module !
30 Not a lock !
31 Bad History value (2..1000)
32 Error opening trackdisk device !
33 DoIO returned with a non zero value !
34 There is no task to debug !
35 Unknown argument for 'trace'!
36 Unknown argument for 'debug'!
37 Address is in ROM ! Can't set breakpoint !
38 Not a debug node !
39 Bad '@' argument
40 LoadSeg error !
41 Unknown argument for 'dmode'!
42 There is no current debug task !
43 Unknown argument for 'break'!
44 Breakpoint does not exist !
45 There are no symbol hunks !

```
46   Symbol not found !
47   You can only remove variables !
48   You can not assign to a function !
49   You must use brackets with functions !
50   Unknown argument for 'symbol'!
51   There are no symbols !
52   You can only realloc blocks smaller than 64K !
53   Your brackets are really out of order !
54   The debug task is busy, please try again later !
55   The task is not tracing !
56   File does not have the right format !
57   This is not a structure definition !
58   Bad argument value !
59   No colorchange allowed when PowerVisor is on a window !
60   Could not open font !
61   Can't execute script file in script file !
62   Refresh window is not open !
63   Unknown tag type !
64   No output allowed on debug logical window !
65   Error writing file !
66   Not a Tag file !
67   Bad tag list value (0 .. 15) !
68   Unknown or invalid register !
69   There is a more recent patch then this one installed !
70   There is no fd file loaded for this library !
71   Unknown logical window !
72   Resulting commandline too long after converting from alias string !
73   Missing left bracket '(' in fd file !
74   Missing right bracket ')' in fd file !
75   Bad '##bias' statement in fd file !
76   You can't close the 'Main' physical window !
77   You can't close the 'Main' logical window !
78   Bad argument for 'openlw' ! Arg must be one of 'r','l','d' or 'u' !
79   Brother logical window must be on same physical window !
80   Window is not movable !
81   There is no father for this box !
82   Unknown preferences argument !
83   Please close visitor windows first !
84   Divide by zero !
85   No group operators allowed for the debug task !
86   Error opening screen !
87   You can't remove the 'rc' and 'error' variables. These are private !
88   'LoadSeg' failed !
89   Variable names must start with a letter or an underscore !
90   Error opening physical window !
91   Error opening logical window !
92   Bracket ')' expected !
93   Bracket '}' expected !
94   Bus error !
95   Address error !
96   Illegal instruction !
97   Division by zero !
98   CHK instruction !
99   TRAPV instruction !
100  Privilege Violation !
101  Trace error !
102  Unimplemented 1010 opcode !
```

```
103  Unimplemented 1111 opcode !
104  Unknown argument for 'track'!
105  PowerVisor is allready tracking for a task !
106  PowerVisor is not tracking !
107  Unknown argument for 'source'!
108  There are no debug hunks in this executable file !
109  There is no source loaded for this debug node !
110  This address is not in the source !
111  This command only works on the 68030 or 68040 !
112  You need a MMU for this command !
113  Not a valid executable file!
114  This operation is not allowed for slave instances!
115  First install memory protection system with 'watch'!
116  You can't freeze PowerVisor!
117  You can't change the protection use of tag list 0! It is global!
118  Unknown argument for 'struct'!
119  Bad internal structure format!
120  This is a read-only structure! You can't change its fields!
121  I can't find this field in the structure!
122  Unknown argument for 'prof'!
123  You are not profiling!
124  You are already profiling!
125  Not a logical window!


Related commands:  error

Related tutor chapters:  The wizard corner
```

1.18 Function Reference : getline()

```
   <pointer to line> = GETLINE( )


Get the pointer to the line at the current screenposition in the current
logical window. This pointer points just after the attribute (hilighted
or not). Note that this line is not null-terminated. You can use the
 string  command to get a null-terminated line.


Related commands:  locate   home   cls   current

Related functions:  getx()   gety()   getcols()   getrows()   cols()
                    lines()   getlwin()   getchar()

Related lists:  lwin

Related tutor chapters:  Screens and windows
```

1.19 Function Reference : getlwin()

```
   <logwin> = GETLWIN( )
```

This function returns a pointer to the current logical window.
The current logical window is not the same as the active logical window.
The current logical window is the one that shows all output. The active
logical window is the one where you can scroll with the keyboard.

Related commands: current on active

Related functions: getactive()

Related lists: lwin

Related tutor chapters: Screens and windows

1.20 Function Reference : getmmuentry()

```
   <address> = GETMMUENTRY( <address> )
```

(only 68020 with 68851 or 68030)
This function returns the address of the MMU entry used to describe the
given address in the MMU tree. This function returns 0 if the MMU is not
used or if there is no MMU entry for the given address.

Related commands: mmutree watch protect

1.21 Function Reference : getrow()

```
   <rows> = GETROW( <logical window> )
```

Get the real number of rows (or -1) for the logical window. See the
explanation for getcol() to see what the -1 really means.

'getrow' uses autodefault to the 'lwin' list for the first argument.

Related commands: colrow fit

Related functions: getcol() cols() lines() getlwin() getx()
gety()

Related lists: lwin

Related tutor chapters: Screens and windows

1.22 Function Reference : getsize()

```
   <size> = GETSIZE( <memoryblock> )
```

This function determines the size of a memoryblock allocated with the alloc() function or some other PowerVisor commands.

Related commands: cleanup showalloc

Related functions: alloc() free() isalloc() realloc()

1.23 Function Reference : getstack()

```
   <max stack> = GETSTACK( )
```

This function returns the maximum stack usage for the last task monitored with the stack command. If the stack monitor is still active, this number will be EXACTLY equal to the maximum stack usage. If the stack monitor is not active any more, this number will be an approximation depending on the last usage of 'getstack' and the number of microseconds with the 'stack' command. See the 'stack' command for an explanation of the algorithm.

Related commands: stack

1.24 Function Reference : getsymstr()

```
   <pointer to string> = GETSYMSTR( <address> )
```

Return the pointer to the string for the symbol on <address>. If there is no symbol on <address>, this function returns 0.
This function returns a string if used from ARexx.

Related commands: debug

1.25 Function Reference : getx()

```
   <current x position> = GETX( )
```

Get the current x coordinate on the current logical window (in columns).

```
Related commands:  locate   home   cls   current

Related functions:  gety()   getcols()   getrows()   cols()   lines()
                    getlwin()   getchar()

Related lists:  lwin

Related tutor chapters:  Screens and windows
```

1.26 Function Reference : gety()

```
   <current y position> = GETY( )


Get the current y coordinate on the current logical window (in lines).


Related commands:  locate   home   cls   current

Related functions:  getx()   getcols()   getrows()   cols()   lines()
                    getlwin()   getchar()

Related lists:  lwin

Related tutor chapters:  Screens and windows
```

1.27 Function Reference : if()

```
   <result> = IF( <condition>,<expression 1>,<expression 2> )


If <condition> is true, evaluate <expression 1>, else evaluate
<expression 2>.


Example :

< d if(1,2,3) <enter>
> 00000002 , 2

< d if(0,2,3) <enter>
> 00000003 , 3

< a=1 <enter>
< b=2 <enter>
< c=3 <enter>
< d=4 <enter>
< e=5 <enter>
< d if(a,if(b,c,d),e) <enter>
> 00000003 , 3

< void if(a==1,{help functions},{help commands}) <enter>
```

```
> ...

Related commands:  disp   void

Related functions:  eval()

Related tutor chapters:  Expressions
```

1.28 Function Reference : isalloc()

```
   <pointer to pointer> = ISALLOC( <pointer> )

This function checks if <pointer> points to a memoryblock allocated
with the  alloc()  function. If it is, 'isalloc' returns a pointer to
the pointer, otherwise 0.

Related commands:  cleanup   showalloc

Related functions:  alloc()   free()   realloc()   getsize()
```

1.29 Function Reference : isbreak()

```
   <result> = ISBREAK( <address> )

Test if there is a breakpoint in the current debug task on <address>.
If true this function returns the breakpoint number and the
breakpoint type in one longword, else it returns 0.

   <result> has the following format :

      NNNNTTTT

   with NNNN a word containing the breakpoint number
   and TTTT containing the breakpoint type

   The following types are supported :
      'T'   temporary breakpoint
      'N'   normal breakpoint
      'P'   profile breakpoint
      'C'   conditional breakpoint
      'A'   after breakpoint (timeout)
      's'   breakpoint used to skip a BSR or JSR (private breakpoint)

Related commands:  debug   break   trace

Related functions:  getdebug()   toppc()   botpc()
```

Related lists: dbug

Related tutor chapters: Debugging

1.30 Function Reference : key()

```
<vanillakey> = KEY( )
```

Wait for a key and return the ascii value.
This is especially useful in scripts.

Related commands: scan request

Related functions: qual()

1.31 Function Reference : lastbytes()

```
<last number of bytes> = LASTBYTES( )
```

This function returns the last number of bytes used by memory or view (320 by default).

Related commands: memory view

Related functions: lastmem() lastlines()

Related tutor chapters: Looking at things

1.32 Function Reference : lastfound()

```
<last found> = LASTFOUND( )
```

This function returns the address of the second byte of the string found with search or next . This is the address where 'next' will continue with the search.

Related commands: search next

Related functions: lastmem()

1.33 Function Reference : lastlines()

```
   <last number of lines> = LASTLINES( )
```

This function returns the last number of lines used by unasm (20 by default).

Related commands: unasm

Related functions: lastmem() lastbytes()

Related tutor chapters: Looking at things

1.34 Function Reference : lastmem()

```
   <memory> = LASTMEM( )
```

This function returns the address that memory , view and unasm will use to continue their listing.

Related commands: memory view unasm

Related functions: lastfound() lastbytes() lastlines()

Related tutor chapters: Looking at things

1.35 Function Reference : lines()

```
   <lines> = LINES( <logwin> )
```

This function returns the maximum number of lines available on the logical window.

'lines' uses autodefault to the 'lwin' list for the first argument.

Example :

```
< disp lines(main) <enter>
> 00000033 , 51
```

Related commands: colrow fit

Related functions: cols() getcols() getrows() getlwin() getx() gety()

Related lists: lwin

Related tutor chapters: Screens and windows

1.36 Function Reference : peek()

```
<value> = PEEK( <structure pointer>,<struct def pointer>,<field name> )
```

Returns the value of <structure>.<field name>. <struct def pointer>
must be a pointer to a previously loaded structure definition. <structure
pointer> is the pointer to the structure itself. <field name> must be
defined in the structure definition.

'peek' uses autodefault to the 'stru' list for the second argument.

Related commands: addstruct interprete struct

Related functions: apeek() stsize()

Related lists: stru

Related tutor chapters: Looking at things

1.37 Function Reference : pubscreen()

```
<pointer to name> = PUBSCREEN( )
```

Returns the pointer to the name of the public screen for this instance of
PowerVisor.
This function returns a string if used from ARexx.

The name of the PowerVisor ARexx port is :

```
PowerVisorScreen
```

or if you are running a slave instance (the first instance of PowerVisor
is the master. All following instances running at the same time with the
master are slaves) :

```
PowerVisorScreen.<num>
```

Related commands: rx

Related functions: arexxport()

Related tutor chapters: Scripts Screens and windows

1.38 Function Reference : qual()

```
   <qualifier> = QUAL( )
```

Return the qualifier for the last pressed key (with key()).

Example :

```
< a=key() <enter>
< disp qual() <enter>
```

Related commands: scan

Related functions: key()

1.39 Function Reference : realloc()

```
   <new pointer> = REALLOC( <memoryblock>,<new size> )
```

Use this function to reallocate a memoryblock allocated with the alloc() function. Only memoryblocks smaller than 64K are supported.

Related commands: cleanup showalloc

Related functions: alloc() free() getsize() isalloc()

1.40 Function Reference : rfrate()

```
   <refresh rate> = RFRATE( )
```

Return the current refresh rate installed with the refresh command.

Related commands: refresh rwin

Related functions: rfcmd()

1.41 Function Reference : rfcmd()

```
   <pointer to refresh command> = RFCMD( )
```

Return a pointer to the current refresh command (or 0).

This function returns a string if used from ARexx.

Related commands: refresh rwin

Related functions: rfrate()

1.42 Function Reference : stsize()

```
<size> = STSIZE( <struct def pointer> )
```

Return the size of the structure represented by <struct def pointer>.
<struct def pointer> must be a structure definition loaded
with addstruct or made with struct .

'stsize' uses autodefault to the 'stru' list for the first argument.

Related commands: addstruct interprete struct

Related functions: peek() apeek()

Related lists: stru

Related tutor chapters: Looking at things

1.43 Function Reference : taglist()

```
<taglist> = TAGLIST( )
```

Return the current tag list. This is a number between 0 and 15.
You can change the current tag list with the usetag or tg commands.

Related commands: usetag tg

Related tutor chapters: Looking at things

1.44 Function Reference : toppc()

```
<programcounter> = TOPPC( )
```

This function returns the program counter visible at the top of the
'Debug' logical window.
You can set this program counter using the dstart or dscroll commands.

Related commands: debug dwin dscroll dstart

Related functions: botpc() isbreak() getdebug()

Related lists: dbug

Related tutor chapters: Debugging