Applixware Generic ELF Reference

This manual was produced using Applixware.

Printed: June 2010

ABS@

Returns the absolute value

Format int = ABS@(value)

Arguments

value	A numeric expression.

Description Returns the absolute value of a numeric expression. If value is negative, ABS@ returns the value (-1 * value). Otherwise, it returns value.

Example

See also INT@

ABSOLUTE_PATH@

Returns an absolute path name

Format path = ABSOLUTE_PATH@(name)

Arguments

name	A string that is the relative name of a file or directory in the current directory.

Description Returns an absolute path name. It returns a value with the resulting path name if name does not exist. If the returned path name contains symbolic links, ABSOLUTE_PATH@ returns the real path, not the link path.

For example, assume that you have a file in ~/macros that is linked to a file in /asterx/axdata/elf. In this case, the root of the returned path name will be /applix/-axdata/elf.

See also CURRENT_DIR@
PARSE_PATHNAME@

ADDDAYS@

Adds or subtracts days from a date number

Format dateValue = ADDDAYS@(dateNumber, numDays)

Arguments dateNumber A serial date number.

numDays The number of days to add or subtract.

Description Adds or subtracts days to a serial date number. For example:

```
ADDDAYS@(33720,7)      returns 33727
ADDDAYS@(33720,-7)     returns 33713
```

You can also use the **DATE@**, **TODAY@** or **NOW@** functions as the first argument in the ADDDAYS@ function, since DATE@, TODAY@, and NOW@ all return serial date numbers. For example, entering the formula ADDDAYS@(DATE@(89,11,9),783) returns the serial date number 33603 which corresponds to January 1, 1991.

ADDHOURS@

Adds or subtracts hours to a serial date or time number

Format ADDHOURS@(dateNumber, numHours)

Arguments dateNumber A serial date number.

numHours The number of hours to add or subtract.

Description Adds or subtracts hours to a serial date or time number. For example:

```
ADDHOURS@(13640,-24)       returns 13639
ADDHOURS@(0.333333333,8)      returns 0.666666666
```

You can also use the **DATE@**, **TODAY@**, **TIME@**, or **NOW@** functions as the first argument in the ADDHOURS@ function. For example, entering the formula ADDHOURS@(DATE@(69,7,20),-48) returns the serial date number 25402 which corresponds to July 18, 1969.

Entering the formula ADDHOURS@(TODAY@(),72) on April 27, 1992 returns the serial date number 33723 which corresponds to April 30, 1992.

Entering the formula ADDHOURS@(TIME@(8,0,0),4) returns the serial time number 0.5 which corresponds to the time 12:00:00 PM.

Entering the formula ADDHOURS@(NOW@(),-36) on April 27,1992 at 12:53:49 PM returns the serial date/time number 33718.995706019 which corresponds to the date April 25,1992 and the time 11:53:49 PM

ADDMINUTES@

Adds or subtracts minutes to a serial date or time number

Format ADDMINUTES@(dateNumber, numMinutes)

Arguments

dateNumber	A serial date number.
numMinutes	The number of minutes to add or subtract.

Description Adds or subtracts minutes to a serial date or time number. For example:

```
ADDMINUTES@(33000,-1440)  returns 32999
ADDMINUTES@(0.5,180)   returns 0.625
```

You can also use the **DATE@**, **TODAY@**, **TIME@**, or **NOW@** functions as the first argument in the ADDMINUTES@ function. For example entering the formula ADDMINUTES@(DATE@(14,6,28),3000) returns the serial date/time number 5294.083333333 which corresponds to the date June 30, 1914 and the time 2:00:00 AM.

Entering the formula ADDMINUTES@(TODAY@(),2160) on April 27, 1992 returns the serial date/time number 33721.5 which corresponds to the date April 28, 1992 and the time 12:00:00 PM.

Entering the formula ADDMINUTES@(TIME@(18,0,0),45) returns the serial time number 0.78125 which corresponds to the time 6:45:00 PM.

Entering the formula ADDMINUTES@(NOW@(),60) on April 27,1992 at 1:41:54 PM returns the serial date/time number 33720.612430556 which corresponds to the date April 27,1992 and the time 2:41:54 PM.

ADDMONTHS@

Adds or subtracts months to a serial date number

Format ADDMONTHS@(dateNumber, numMonths)

Arguments

dateNumber	A serial date number.
numMonths	The number of months to add or subtract.

Description Adds or subtracts months to a serial date number.

```
ADDMONTHS@(31000,12)      returns 31365
ADDMONTHS@(30000,-6)      returns 29816
```

You can also use the **DATE@**, **TODAY@**, or **NOW@** functions as the first argument in the ADDMONTHS@ function. For example, entering the formula ADDMONTHS@(DATE@(76,6,1),18) returns the serial date number 28459 which corresponds to the date December 1, 1977.

Entering the formula ADDMONTHS@(TODAY@(),61) on April 28, 1992 returns the serial date number 35577 which corresponds to the date May 28, 1997.

Entering the formula ADDMONTHS@(NOW@(),15) on April 28, 1992 at 10:33:16 AM returns the serial date/time number 34177.439768519 which corresponds to the date July 28, 1993 and the time 10:33:16 AM.

ADDSECONDS@

Adds or subtracts seconds to a serial date or time number

Format ADDSECONDS@(dateNumber, numSeconds)

Arguments dateNumber A serial date number.

numSeconds The number of seconds to add or subtract.

Description Adds or subtracts seconds to a serial date or time number. For example:

ADDSECONDS@(0.333333333,90) returns 0.334375

You can also use the **DATE@**, **TODAY@**, **TIME@** or **NOW@** functions as the first argument for the ADDSECONDS@ function. For example, entering the formula ADDSECONDS@(DATE@(12,4,14),3727) returns the serial date/time number 4487.043136574 which corresponds to the date April 14, 1912 and the time 1:02:07 AM.

Entering the formula ADDSECONDS@(TODAY@(),777) on April 28, 1992 returns the serial date/time number 33721.008993056 which corresponds to the date April 28, 1992 and the time 12:12:57 AM.

Entering the formula ADDSECONDS@(TIME@(3,0,15),22) returns the serial time number 0.125428241 which corresponds to the time 3:00:37 AM.

Entering the formula ADDSECONDS@(NOW@(),33) on April 28, 1992 at 11:34:12 AM returns the serial date/time number 33721.482465278 which corresponds to the date April 28, 1992 and the time 11:34:45 AM.

ADDYEARS@

Adds or subtracts years to a serial date number

Format ADDYEARS@(dateNumber, numYears)

Arguments

dateNumber	A serial date number.
numYears	The number of years to add or subtract.

Description Adds or subtracts years to a serial date number. For example:

ADDYEARS@(33721,5) returns 35547
ADDYEARS@(33721,-5) returns 31894

You can also use the **DATE@**, **TODAY@**, or **NOW@** functions as the first argument for the ADDYEARS@ function.

For example, entering the formula ADDYEARS@(DATE@(88,1,19),4) returns the serial date number 33621 which corresponds to the date January 19, 1992.

Entering the formula ADDYEARS@(TODAY@(),5) on April 28, 1992 returns the serial date number 35547 which corresponds to the date April 28, 1997.

Entering the formula ADDYEARS@(NOW@(),-37 on April 28, 1992 at 11:59:47 AM returns the serial date/time number 20206.499849537 which corresponds to the date April 28, 1955 and the time 11:59:47 AM.

ALL_WINDOWS_BUSY@

Posts an hourglass cursor

Format ALL_WINDOWS_BUSY@()

Description Changes the cursor's shape to an hourglass. Programs should do this to indicate that the program is busy and cannot accept input. When your program becomes ready to receive data, ELF automatically changes the cursor back to its normal shape.

ALT_LANG_DIR@

Returns the language directory

Format dir = ALT_LANG_DIR@()

Description Returns the name of the directory that contains the Applixware-provided language and hyphenation dictionaries.

APP_IS_DATA@

Returns task information if application is Data

Format infoArray = APP_IS_DATA@()

Description If the current application is Applixware Data, returns a one-element array whose value is TRUE. Otherwise, it returns a three-element array whose contents are as follows:

infoArray[0]	TRUE, indicating that the current application is data.
infoArray[1]	The task id of the current application's parent. That is, this is the **MACRO_PARENT_TASK@** id.
infoArray[2]	The current task's task id. That is, this is the **ELF_TASK_ID@**.

ARRAY_APPEND@

Appends one array onto the end of another

Format array = ARRAY_APPEND@(to, from)

Arguments

to	The array you are appending to.
from	The array you are appending from.

Description Appends the from array onto the end of the to array. The new to array is returned. For example, if to has 5 elements and from has 2 elements, the returned array has 7 elements.

Example

See also **ARRAY_TRANSPOSE@**
FORMAT_ARRAY@
SORT@

ARRAY_COLUMN@

Extracts one column from a 2D array

Format col = ARRAY_COLUMN@(array, colNum)

Arguments

array	A two-dimensional ELF array.
colNum	The column of data to be extracted. This number is zero-based.

Description Creates a one-dimensional array whose values are those contained in one column of an ELF two-dimensional array.

For example, assume you have an 3 x 4 array. To extract elements [0,2], [1,2], and [2,2], you'd invoke ARRAY_COLUMN@(array,2). Because the column numbers are zero-based, the value 2 is extracting the *third* array column.

Example

ARRAY_DELETE@

Removes an item from an array

Format array = ARRAY_DELETE@(array, index)

Arguments

array	The array from which you are deleting an element (or subarray).
index	The portion of the element or sub-array within array which is being deleted.

Description Removes an item from an array. array [index] is deleted with items after this location moving down by one. The resulting array is returned; array is not modified in place.

Example

See also ARRAY_INDEX@
ARRAY_INSERT@
SUBARRAY_REMOVE@
LIST_REMOVE@

ARRAY_FROM_ARGS@

Converts passed argument to an array

Format array = ARRAY_FROM_ARGS@(...)

Description Converts passed arguments into an array. This macro only converts passed arguments, not declared arguments. Therefore, if no arguments are passed, but five are declared, the array size is zero. The inverse is also true.

See also ARRAY_FROM_STRING@

ARRAY_FROM_STRING@

Returns an array representing the passed string

Format array = ARRAY_FROM_STRING@(string, sepChar)

Arguments

string	The passed string.
sepChar	A character used to separate each "word" in the string.

Description Returns an array representing the passed string. Each "word" in the string is returned as a separate array element. "Words" are separated by the passed sepChar.

The words will be trimmed unless the sepChar is a space. A null array is returned if string is empty. For example, "The Quick Brown Fox" is returned as a four element array. "Quick" is the second element of this array.

Example

See also ARRAY_FROM_ARGS@
COMMA_SPLIT@

ARRAY_INDEX@

Searches an array for a string or number

Format pos = ARRAY_INDEX@(array, value)

Arguments

array	The array to search; array can contain strings, numbers, or sub-arrays.

value	The value to be located.

Description Searches an array for the string or number indicated by value. If value is found, ARRAY_INDEX@ returns value's array position. If value is not found, ARRAY_INDEX@ returns -1. Because arrays are zero-based, the value returned is the array's index position. For example, if value is in array[2], the returned value is *2*, which indicates that the value is in the array's third element.

ARRAY_INDEX@ works only with single-dimensional arrays. You can't pass ARRAY_INDEX@ a multi-dimensional array. You can pass ARRAY_INDEX@ a single-dimensional array that is a portion of a mullti-dimensional array. The following example shows this:

```
macro test

   var month_array, position

   month_array[0] = "x"
   month_array[1] = "jan"
   month_array[2] = "feb"
   month_array[3] = "mar"
   month_array[4] = "apr", "Apr"
   month_array[5] = "may"
   month_array[6] = "jun"
   month_array[7] = "jul", "Jul"
   month_array[8] = "aug"
   month_array[9] = "sep"
   month_array[10] = "oct"
   month_array[11] = "nov"
   month_array[12] = "dec"

   DUMP_ARRAY@( month_array )


/*  position = ARRAY_INDEX@( month_array, "Apr" )
        This doesn't work. (month_array[] is multi-dimensional)   */
   position = ARRAY_INDEX@( month_array[4], "Apr" )
        /* This works  */

   INFO_MESSAGE@( position )

endmacro
```

Example

See also ARRAY_DELETE@
ARRAY_INSERT@
SUBARRAY@

ARRAY_INSERT@

Inserts an item into an array

Format newArray = ARRAY_INSERT@(array, item, index)

Arguments

array	The array into which you are inserting item.
item	The item that you are inserting into array. This item can be a single element, such as a string or interger, or an array.
index	The location in array into which you are inserting item.

Description Inserts item into array at position index. Before item is inserted, all elements from array[index] to the end of the array are moved up one position. The resulting array, newArray, is returned. array is not modified in place.

ARRAY_INSERT@ always inserts the item into the first dimension of the target array, even if the target array has more than one dimension. For example, suppose you have a two dimensional array of strings, like this:

x[0] = "hello", "world", "my", "name", "is", "Robert"

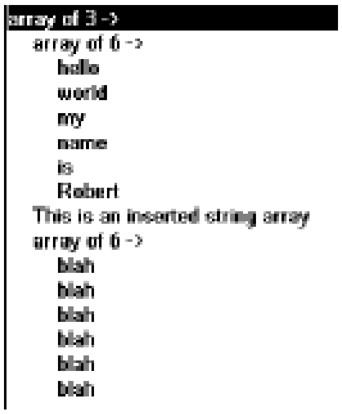
x[1] = "blah", "blah", "blah","blah","blah","blah"

If you execute these two statements:

y = "This is an inserted string array"

z = array_insert@(x, y, 1)

The resulting array z looks like this:



```
array of 3 ->
    array of 6 ->
        hello
        world
        my
        name
        is
        Robert
    This is an inserted string array
    array of 6 ->
        blah
        blah
        blah
        blah
        blah
        blah
```

Note that the string Y is inserted into the first dimension of the array, and not into either of the two sub-arrays.

Example

See also ARRAY_DELETE@
ARRAY_INDEX@
ARRAY_INSERT_ORDERED@

ARRAY_INSERT_ORDERED@

Inserts an item into an ordered (sorted) array

Format newArray = ARRAY_INSERT_ORDERED@(array, item[, caseFlag[, duplicateFlag]])

Arguments	
array	The array into which you are inserting item.
item	The item that you are inserting into the array.
caseFlag	A Boolean value, where TRUE means ignore case when inserting item in an array of strings; FALSE means insert by case.
duplicateFlag	A Boolean value, where TRUE means allow duplicate entries in the array; FALSE means do not allow duplicates.

Description Inserts item into array in numeric or alphabetical order.

Example

See also ARRAY_DELETE@
ARRAY_INDEX@

ARRAY_INSERT@

ARRAY_SIZE@

Determines the number of elements in an array

Format size = ARRAY_SIZE@(array)

Arguments array The name of the array whose size you are determining.

Description Returns the number of elements in an array.

Because ELF arrays are zero-based (that is, the first element is element 0), this number is one more than the index value you would use to access the last element in array. For example, to access the last item in a 40-element, you would use an array index of 39. In the following example, notice how the size of the array is decremented by 1:

```
for i=0 to (ARRAY_SIZE@(anArray)-1)
        ...
next i
```

Example

ARRAY_TO_BINARY@

Converts an ELF array to a binary object

Format binArray = ARRAY_TO_BINARY@(array)

Arguments array The ELF array being converted into a binary object. The array must contain only small (byte-size) numbers

Description The converted binary array is returned. Typically, ARRAY_TO_BINARY@ converts an ELF array created using BINARY_TO_ARRAY@.

See also ARRAY_TO_STRING@

ARRAY_TO_STRING@

Returns a string representing the passed array

Format string = ARRAY_TO_STRING@(array, sepChar)

Arguments

array	The name of the array.
sepChar	The character that separates each "word" in the string.

Description Returns a string constructed from the passed array. Each member in the array is returned as a "word" in the string. "Words" are separated by the passed separation character.

See also ARRAY_TO_BINARY@
COMMA_SEPARATE@

ARRAY_TRANSPOSE@

Transposes an array

Format newArray = ARRAY_TRANSPOSE@(array)

Arguments

array	The array being transposed.

Description Returns an array that is the transpose of another array. For example, if array is a 4 x 3 array, the new transposed array is a 3 x 4. This macro will not work on a 1-dimensional array. If you use a 1-dimensional array as an argument, ARRAY_TRANSPOSE@ will throw an error (Not an Array).

See also ARRAY_APPEND@
FORMAT_ARRAY@
SORT@

ASSIGN_GRAPHIC@

Assigns graphics memory to a different task

Format ASSIGN_GRAPHIC@(gfx, taskId)

Arguments

gfx	Initially, the old graphics handle. After this macro executes, it contains a handle to the assigned graphic editor.
taskId	An Applix*ware* task ID number.

Description Assigns the graphic task indicated by gfx to task taskid. This macro changes the task's owner from its current task to a different task. The handle to this new graphics task is returned by changing the value of gfx to this new value.

AX_APP_TYPE_FROM_DOC_TYPE@

Returns a constant indicating which Applixware document type is being tested

Format appType = AX_APP_TYPE_FROM_DOC_TYPE@ (docType)

Arguments docType A document type value. Typically, this value is returned by **RECOGNIZE_FILE@**

Description Returns one of the following constants:

AX_APP_TYPE_AUDIO
AX_APP_TYPE_EQUATION
AX_APP_TYPE_NONE
AX_APP_TYPE_GRAPHICS
AX_APP_TYPE_SPREADSHEET
AX_APP_TYPE_WORDS

AX_DOC_TYPE_IS_NATIVE@

Returns a Boolean indicating if argument is an Applixware document type

Format flag = AX_DOC_TYPE_IS_NATIVE@(docType)

Arguments docType A document-type number. Typically, this value is returned by **RECOGNIZE_FILE@**.

Description Returns a Boolean value which if TRUE indicates that the document is an Applix*ware* document.

AX_GET_SHARED_LIB_INFO@

Returns info on shared libraries loaded in Applixware

Format arrays = AX_GET_SHARE_LIB_INFO@()

Description Returns a set of four-element arrays. One array is returned for every shared library currently loaded in Applix*ware*. Each array contains the following information:

- The absolute path of the shared library.

- The name of the macro or C function that is first run when the shared library is called.
- A boolean. If TRUE (-1), the first function is an ELF macro. If FALSE (0), the first function is a C function.
- A raw time stamp indicating the time when the shared library was loaded.

ax_inset@ format

Format that defines an inset

Description The ax_inset@ format is defined in the ELF include file insets_.am. This format is used by ELF macros that create and manipulate insets within dialog boxes and Applix*ware* documents. The data in an ax_inset@ format consists of one ax_inset_object@ format and one ax_inset_info@ format. The fields in these format are as follows:

format ax_inset@

```
        format ax_inset_object@ obj,
        format ax_inset_info@ info
```

format ax_inset_object@

```
        type,         ' type of object, as defined in app_ids_.am:
                      ' APP#WORDS_, APP#GRAPHICS_, APP#SPREADSHEET_, etc.
        uid,          ' type specific unique identifier, as follows:
                      ' wpx          a Words inset
                      ' gfx          a Graphics inset
                      ' hmx          an HTML inset
        info,         ' Opaque, type-specific structure; null -> default
        units,        ' this will override units in opaque source structure
        parents               ' array of parents - reserved
```

format ax_inset_info@

```
        format ax_units@ units,
        format ax_area@ destination,
        format ax_area@ clip,
        suppress_cursor,    ' true  - don't change pointer while rendering (default)
                            ' false - the renderer may change the pointer
        direct_render,              ' true  - draw directly onto surface
                            ' false - draw offscreen, copy to surface (default)
        icon_only,      '           true  - don't draw the source, just an icon
                            ' false - draw the source material (default)
```

```
x_zoom,                         ' any number - 100.0 is 100% or 1X zoom (default)
y_zoom,                 ' any number - 100.0 is 100% or 1X zoom (default)
scale_mode,             ' 0 / "clip to fit"    - no scaling (excluding zoom)
                        ' 1 / "scale to fit"   - scale to fit destination (default)
                        ' 2 / "scale if clips" - if the rendering is too big
                        ' to fit into the destination, the rendering should be
                        ' scaled to fit
proportional,  ' true  - scaling may NOT change aspect ratio (default)
               ' false - scaling may distort aspect ratio
centered,      ' true  - source is centered in destination
               ' false - source is aligned to upper left corner (default)

widget_colors,          ' true  - use widget color for screens and clears
               ' false - use window colors (default)
clear,         ' true  - clear the background before rendering (default)
               ' false - do not clear the background before rendering
fill,          ' true  - fill the background before rendering
               ' false - don`t fill the backgrnd before rendering(default)
border_style, ' 0 / "none"         - no border (default)
               ' 1 / "solid"        - solid border
               ' 2 / "dashed"        - dashed border
inactive_content,       ' true  - overlay a 50% screen after rendering
                        ' false - do not overlay a 50% screen (default)
inactive_border,                ' true  - overlay a 50% screen over border
                        ' false - do not overlay a 50% screen (default)
```

AX_MS@

Indicates if Applixware is running under Windows 95 or Windows NT

Format flag = AX_MS@()

Description Returns a Boolean value where TRUE indicates that Applix*ware* is running either under Windows NT or Windows 95.

AX_NT@

Indicates if Applixware is running under Windows NT

Format flag = AX_NT@()

Description Returns a Boolean value where TRUE indicates that Applix*ware* is running under Windows NT.

AX_WIN95@

Indicates if Applixware is running under Windows 95

Format flag = AX_WIN95@()

Description Returns a Boolean value where TRUE indicates that Applix*ware* is running under Windows 95.

AXHOME_DIR@

Returns the location of the user's axhome directory

Format dir = AXHOME_DIR@()

Description Returns the location (path name) of the user's axhome directory. For example: /user/john_doe/axhome.

Example

See also SYSTEM_DIR@

AXTRACE_OFF@

Disables External Function Call Tracing in the Spreadsheet

Format AXTRACE_OFF@

Description Turns off external function call tracing in the Spreadsheet and closes the log file opened by the AXTRACE_ON@ macro.

See also AXTRACE_ON@

AXTRACE_ON@

Enables External Function Call Tracing in the Spreadsheet

Format AXTRACE_ON@(LogtFileName)

Description Turns on external function call tracing in the Spreadsheet and writes logging information to the specified log file name.

Arguments LogFileName A string containing the name of the file in which to write the trace information.

See also AXTRACE_OFF@

BACKLIT_TEMPLATE_DIR@

Returns the backlit template directory

Format BACKLIT_TEMPLATE_DIR@()

Description Returns the directory containing the backlit color slide templates used by Applixware Presents.

BEEP@

Sounds the system bell

Format BEEP@()

Description On a Windows workstation, BEEP@ uses the control panel setting for default sound. Choose Control Panel -> Sounds -> Default Sound to change the setting.

Example

BINARY_BYTE@

Returns a byte value within a binary object

Format byte = BINARY_BYTE@(object, offset)

Arguments	object	A binary data object.
	offset	The location of the defined byte within object.

Description Bytes are numbered from zero on up. BINARY_BYTE@ is useful for reading the bytes from a binary object one byte at a time.

See also BINARY_SIZE@

BINARY_SIZE@

Returns the number of bytes in a binary object

Format size = BINARY_SIZE@(object)

Arguments	object	A binary data object.

Description Returns a number indicating the number of bytes in a binary object.

See also BINARY_BYTE@

BINARY_TO_ARRAY@

Converts a binary object into an ELF array

Format array = BINARY_TO_ARRAY@(object, level)

Arguments	object	The binary data object to convert.
	level	The maximum depth for each sub-array in an array.

Description Returns a two-dimensional array of the bytes in a binary object. Each array element is an array of the length specified by level. For example, specifying a level of 4, places the byte of a word into a subarray, so that word[0] is in element array [0,0], [0,1], [0,2] and [0,3]. BINARY_TO_ARRAY@ provides an easy way to manipulate individual bytes in the binary object.

See also ARRAY_TO_BINARY@

BINARY_TO_STRING@

Converts a binary object into an ELF string

Format string = BINARY_TO_STRING@(object)

Arguments object — The binary data object to convert.

Description Converts a data object from binary format to a text string.

See also STRING_TO_BINARY@

BIT7_ASCII_TO_STRING@

Transforms input string to 8-bit clean output string

Format length = BIT7_ASCII_TO_STRING@(in, out, maxSize, flag)

Arguments

in	The string being converted.
out	The converted output string.
maxSize	The maximum number of characters that out can contain.
flag	A Boolean value which if set to TRUE indicates that backslash characters are used as an ESCAPE character within the in string.

Description Converts an external string into internal format.

BIT8_?@

Places the specified character at the current cursor location

Format BIT8_?@()

Description All BIT8_?@ macros place a character at the current cursor location in the current document. For brevity, all the BIT8_?@ macros are listed in this entry. The BIT8_?@ macros and the characters they produce are as follows:

bit8_cent@ - ¢	bit8_sterling@ - £	bit8_brokenbar@ - ¦
bit8_eclamdown@ - ¡	bit8_yen@ - ¥	bit*_ccedilla@ - ç

bit8_section@ - §
bit8_dieresis@ - ¨
bit8_copyright@ - ©
bit8_currency@ - ¤
bit8_ordfeminine@ - ª
bit8_guillemotleft@ - «
bit8_logicalnot@ - ¬
bit8_hyphen@ - -
bit8_registered@ - ®
bit8_macron@ - ¯
bit8_degree@ - °
bit8_plusminus@ - ±
bit8_questiondown@ - ¿
bit8_twosuperior@ - ²
bit8_threesuperior@ - ³
bit8_acute@ - ´
bit8_mu@ - µ
bit8_paragraph@ - ¶
bit8_periodcentered@ - ·
bit8_cedilla@ - ¸
bit8_onesuperior@ - ¹
bit8_ordmasculine@ - º
bit8_guillemotright@ - »
bit8_onequarter@ - ¼
bit8_onehalf@ - ½
bit8_threequarters@ - ¾
bit8_A_grave@ - À
bit8_A_acute@ - Á
bit8_A_circumflex@ - Â
bit8_A_dieresis@ - Ã
bit8_A_tilde@ - Ä
bit8_A_ring@ - Å
bit8_A_E@ - Æ
bit8_C_cedilla@ - Ç
bit8_E_grave@ - È
bit8_E_acute@ - É
bit8_E_circumflex@ - Ê
bit8_E_dieresis@ - Ë
bit8_I_grave@ - Ì
bit8_I_acute@ - Í
bit8_I_circumflex@ - Û
bit8_I_dieresis@ - Ï
bit8_E_th@ - Ð
bit8_N_tilde@ - Ñ
bit8_O_grave@ - Ò
bit8_O_acute@ - Ó
bit8_O_circumflex@ - Ô
bit8_O_tilde@ - Õ
bit8_O_dieresis@ - Ö
bit8_multiply@ - ×
bit8_O_slash@ - Ø
bit8_U_grave@ - Ù
bit8_U_acute@ - Ú
bit8_U_circumflex@ - Û
bit8_U_dieresis@ - Ü
bit8_Y_acute@ - Ý
bit8_T_horn@ - Þ
bit8_germandbls@ - ß
bit8_agrave@ - à
bit8_aacute@ - á
bit8_acircumflex@ - â
bit8_atilde@ - ã
bit8_adieresis@ - ä
bit8_aring@ - å
bit8_ae@ - æ
bit8_ccedilla@ - ç
bit8_egrave@ - è
bit8_eacute@ - é
bit8_ecircumflex@ - ê
bit8_edieresis@ - ë
bit8_eth@ - ð
bit8_e_th@ - Ð
bit8_igrave@ - ì
bit8_iacute@ - í
bit8_icircumflex@ - î
bit8_idieresis@ - ï
bit8_eth@ - ð
bit8_ntilde@ - ñ
bit8_ograve@ - ò
bit8_oacute@ - ó
bit8_ocircumflex@ - ô
bit8_otilde@ - õ
bit8_odieresis@ - ö
bit8_divide@ - ÷
bit8_oslash@ - ø
bit8_ugrave@ - ù
bit8_uacute@ - ú
bit8_ucircumflex@ - û
bit8_udieresis@ - ü
bit8_yacute@ - ý
bit8_thorn@ - þ
bit8_t_horn@ - Þ
bit8_ydieresis@ - ÿ

BUILDER_APPLICATION_DLG@

Starts Applixware Builder

Format taskID = BUILDER_APPLICATION_DLG@([ObjectName])

Arguments ObjectName The name of the object to open.

Description Starts the Builder programming environment. The ObjectName parameter allows you to name the default object that appears on the screen when Builder initializes.

BUILDER_APPLICATION_DLG@ returns the Builder task ID.

BUILD_PATHNAME@

Creates a pathname from a directory and a file name

Format BUILD_PATHNAME@(dir, filename, path)

Arguments

dir	The name of a directory
filename	the name of a file
path	an output parameter that contains the path created from the file name and the directory. Set this parameter to NULL before you call BUILD_PATHNAME@.

Description Constructs a pathname from the file name and directory. If the supplied directory does not contain a trailing slash, a slash is inserted before the file name. However, no slash is inserted at the beginning of the pathname. The following example macro prints the name directory/filename on the screen.

```
macro test
var dir, file, path
        dir = "directory"
        file = "filename"
        path = NULL
BUILD_PATHNAME@(dir, file, path)
info_message@(path)
endmacro
```

BYTE_LEN@

Returns the number of bytes in a string

Format BYTE_LEN@(string)

Arguments

string	a string of characters. These can be multi-byte characters.

Description Returns the number of bytes in a string of characters. Note that in oriental languages, the number of bytes is not equal to the number of characters. In latin-based languages, the number of bytes in a string is equal to the number of characters in the string.

See also LEN@

Document Type Codes

AX_DOC_TYPE_ASCII	0	/* ASCII */
AX_DOC_TYPE_APPLIX_MARKUP	2	/* Applixware Markup */ (NOT USED)
AX_DOC_TYPE_WP_51	5	/* Word Perfect 5.1 */
AX_DOC_TYPE_MS_WORD	6	/* MS Word */
AX_DOC_TYPE_OFFICE_WRITER	7	/* OfficeWriter */
AX_DOC_TYPE_WP_42	8	/* Word Perfect 4.2 */
AX_DOC_TYPE_SAMNA	9	/* Samna */
AX_DOC_TYPE_VOLKS_WRITER	10	/* VolksWriter */
AX_DOC_TYPE_LEADING_EDGE	11	/* Leading Edge */
AX_DOC_TYPE_WORDSTAR	12	/* WordStar */
AX_DOC_TYPE_WORDSTAR_2000	13	/* WordStar2000 */
AX_DOC_TYPE_XYWRITE	14	/* XYwrite */
AX_DOC_TYPE_PFS_PROFESSIONAL	15	/* PfsProfessional */
AX_DOC_TYPE_PFS_WRITE	16	/* PfsWrite */
AX_DOC_TYPE_EXECUTIVE_WRITER	17	/* ExecutiveWriter */
AX_DOC_TYPE_SPELL_BINDER	18	/* SpellBinder */
AX_DOC_TYPE_PC_WRITE	19	/* PCwrite */
AX_DOC_TYPE_Q_AND_A	20	/* QandA */
AX_DOC_TYPE_MULTI_MATE	21	/* MultiMate */
AX_DOC_TYPE_MANUSCRIPT	22	/* Manuscript */
AX_DOC_TYPE_SUN_WRITE	23	/* SunWrite */
AX_DOC_TYPE_DISPLAY_WRITE	25	/* DisplayWrite */
AX_DOC_TYPE_RTF	26	/* Rich Text Format (RTF) */
AX_DOC_TYPE_MS_WFW2	27	/* MS Word for Windows */
AX_DOC_TYPE_ASTERX2	29	/* Applix3 --> aster*x2.1 */
AX_DOC_TYPE_WINWORD60	34	/* Win Word 6.0 */
AX_DOC_TYPE_HTML	35	/* HTML for World Wide Web */
AX_DOC_TYPE_XWD	51	/* XWD */
AX_DOC_TYPE_FAX_M	52	/* FAX msb */
AX_DOC_TYPE_EPSI	53	/* EPSI */
AX_DOC_TYPE_HPGL	54	/* HPGL */
AX_DOC_TYPE_X_BITMAP	55	/* XBitmap */
AX_DOC_TYPE_PCX	56	/* PCX */
AX_DOC_TYPE_SUN_RASTER	57	/* Sun raster */
AX_DOC_TYPE_MAC_PAINT	58	/* MacPaint */
AX_DOC_TYPE_MS_WINDOWS_BITMAP	59	/* MSWindows Bitmap */
AX_DOC_TYPE_CGM	60	/* CGM */
AX_DOC_TYPE_DXF	61	/* DXF */
AX_DOC_TYPE_FAX_I	62	/* FAX lsb */
AX_DOC_TYPE_IGES	63	/* IGES */

AX_DOC_TYPE_TIFF_M	64	/* TIFF M */
AX_DOC_TYPE_TIFF_I	65	/* TIFF I */
AX_DOC_TYPE_GIF	66	/* GIF */
AX_DOC_TYPE_PICT	67	/* PICT */
AX_DOC_TYPE_TGA	68	/* TGA */
AX_DOC_TYPE_ILBM	69	/* ILBM */
AX_DOC_TYPE_GEM	70	/* GEM */
AX_DOC_TYPE_PBM	71	/* PBM */
AX_DOC_TYPE_PGM	72	/* PGM */
AX_DOC_TYPE_PPM	73	/* PPM */
AX_DOC_TYPE_PICT2	74	/* PICT2 */
AX_DOC_TYPE_WMF	75	/* Windows Metafile */
AX_DOC_TYPE_WPG	76	/* Word Perfect Graphics */
AX_DOC_TYPE_GP4	77	/* Group 4 fax */
AX_DOC_TYPE_XPM	78	/* X pixmap */
AX_DOC_TYPE_RAW	79	/* Raw Bitmap */
AX_DOC_TYPE_IRIS	80	/* SGI Iris images */
AX_DOC_TYPE_GR2	82	/* Graphics2.1 */
AX_DOC_TYPE_GR3	83	/* Graphics2.1 */

AX_DOC_TYPE_WKS	100	/* WKS/WK1 */
AX_DOC_TYPE_WK1	100	
AX_DOC_TYPE_SYLK	101	/* SYLK */
AX_DOC_TYPE_DIF	102	/* DIF/XDIF */
AX_DOC_TYPE_XDIF	102	
AX_DOC_TYPE_ASCII_GRID	103	/* ASCII grid */
AX_DOC_TYPE_ASCII_COLUMN	104	/* ASCII into 1 column */
AX_DOC_TYPE_ASCII_ROW	105	/* ASCII into 1 row */
AX_DOC_TYPE_CSV	106	/* CSV */
AX_DOC_TYPE_XLS3	107	/* XLS 3.0 */
AX_DOC_TYPE_XLS4	108	/* XLS 4.0 */
AX_DOC_TYPE_WK3	109	/* WK3 */
AX_DOC_TYPE_ASTERX_SS2	110	/* Applixware Spreadsheets 2.1 */
AX_DOC_TYPE_WK4	111	/* WK4 */

AX_DOC_TYPE_AUDIO	150	/* Audio */
AX_DOC_TYPE_DATA	151	/* Data */
AX_DOC_TYPE_COMPRESSED_AUDIO	152	/* Compressed Audio */

AX_DOC_TYPE_DIRECTORY	160	/* Directory */
AX_DOC_TYPE_CHAR_SPECIAL	161	/* Character special */
AX_DOC_TYPE_BLOCK_SPECIAL	162	/* Block special */
AX_DOC_TYPE_SYMBOLIC_LINK	163	/* Symbolic link */
AX_DOC_TYPE_SOCKET	164	/* Socket */
AX_DOC_TYPE_FIFO	165	/* Fifo */
AX_DOC_TYPE_UNKNOWN	166	/* Unknown */
AX_DOC_TYPE_EMPTY	167	/* Empty */
AX_DOC_TYPE_COMPRESSED_DATA	168	/* Compressed data */
AX_DOC_TYPE_BTOA_DATA	169	/* btoa'd data */
AX_DOC_TYPE_MAC_HEADER	170	/* has Mac Finder header */

AX_DOC_TYPE_ASTERX_WORDS	200	/* Applixware Words */
AX_DOC_TYPE_ASTERX_GRAPHICS	201	/* Applixware Graphics */

AX_DOC_TYPE_ASTERX_SS	203	/* Applixware Spreadsheets */
AX_DOC_TYPE_ASTERX_MACROS	204	/* Applixware Macros */
AX_DOC_TYPE_ASTERX_BINARY_SS	205	/* Binary Spreadsheets */
AX_DOC_TYPE_ASTERX_BITMAP	206	/* Applixware Bitmap */
AX_DOC_TYPE_ASTERX_EQUATIONS	207	/* Applixware Equations */
AX_DOC_TYPE_ASTERX_ASCII	208	/* Applixware external ascii */
AX_DOC_TYPE_ASTERX_BINARY	209	/* Applixware external binary */
AX_DOC_TYPE_ASTERX_ELFARRAY	210	/* Applixware ELF array */
AX_DOC_TYPE_ASTERX_QUERYDATA	211	/* Applixware Data */
AX_DOC_TYPE_ASTERX_MACRO_OBJ	212	/* Applixware Macros */
AX_DOC_TYPE_ASTERX_BUILDER	213	/* Applixware Builder (.ab) */
AX_DOC_TYPE_ASTERX_OBJ_CLASS	214	/* Applixware Builder - Object class library (.ell) */
AX_DOC_TYPE_ASTERX_BUILDER_ABD	215	/* Applixware Builder Distribution (.abd) */
AX_DOC_TYPE_ASTERX_ELF_TABLE	216	/*ELF array in table_data@ format */
AX_DOC_TYPE_ASTERX_BUILDER_TURBO	217	/* Applixware Builder (.abo) */
AX_DOC_TYPE_ASTERX_SLIDESHOW	218	/* Applixware SlideShow (.ago) */

CHANGE_LINKS_FROM_INFO@

Change links based on passed data

Format CHANGE_LINKS_FROM_INFO@(filename, format doc_links_info@ info)

Arguments

filename	The file containing the links.
info	The new link information.

Description Changes the links within filename to the characteristics stated in info. Note that a linked document can also contain links. Using this command, you can change the characteristics of all these links.

You can also use the CHANGE_LINKS_FROM_LIST@ macro to change link information.

For a definition of the doc_links_info@ format, see CHANGE_LINKS_FROM_LIST@.

CHANGE_LINKS_FROM_LIST@

Changes links based on list of links

Format CHANGE_LINKS_FROM_LIST@(filename, format arrayof doc_links_info@ linksList)

Arguments

filename	The file containing the links.

linksList	A flat list containing the link information.

Description Changes the links within filename to the characteristics stated in linksList. Note that a linked document can also contain links. Using this command, you can change the characteristics of all these links.

You can also use the **CHANGE_LINKS_FROM_INFO@** macro to change link information.

The definition of the doc_links_info@ format is as follows:

```
format doc_links_info@
        format arrayof link_info@ links,
        format arrayof object_link_info@ objects
```

The definition of the link_info@ format is as follows:

```
format link_info@
        name,              'Linked file name
        docType,           'Document type returned by RECOGNIZE_FILE_INFO@
        appType,           'Application type returned by RECOGNIZE_FILE_INFO@
        filterMacro,       'Import filter macro
        launchMacro,       'Macro to launch application
        externalRefName,
                           'Reference in linked document
        internalRefName,
                           'Reference in parent document
        format arrayof doc_links_info@ info
```

The definition of the object_link_info@ format is as follows:

```
format object_link_info@
        name,              'Object name
        docType,           'Document type returned by RECOGNIZE_FILE_INFO@
        appType,           'Application type returned by RECOGNIZE_FILE_INFO@
        filterMacro,       'Import filter macro
        launchMacro,       'Macro to launch application
        internalRefName,
                           'Reference in the parent document
        allowUnreferenced,
                           'Boolean: TRUE means save even if not referenced
        format arrayof doc_links_info@
```

CHANGE_PREFS@

Changes a preference setting in the user's ax_prof4 file

Format CHANGE_PREFS@(preference, setting)

Arguments

preference	The preference name in the ax_prof4 file.
setting	The value assigned to the preference.

Description Changes the setting of a specific **preference** in your ax_prof4 file, which is an ASCII file located in your home directory. In many cases, you must log out of Applix*ware* and then back in to invoke the new setting.

If the preference uses more than one argument, you must pass these arguments to ELF as one string. For example:

CHANGE_PREFS@("wpWindowPosSize", "0,0,7800,7800")

Notice the quotation marks surrounding the four window coordinate positions.

To change more than one preference setting at a time, use **EDITPREFS@**. Use **PREF-ERENCES@** to obtain the value of one preference.

Example

CHART_TEMPLATE_DIR@

Returns the chart template directory's name

Format dir = CHART_TEMPLATE_DIR@( )

Description Returns the name of the chart template directory. If this director does not exist, this macro creates it.

This directory is the place in the file system where you will store chart templates and is the place shown when you press the Select Template File button within the Spread-sheets Charts ® Create ® Template command.

CHECK_SPELLING@

Returns an array of misspelled words

Format	wordArray = CHECK_SPELLING@(words)
Arguments	words — The words whose spelling is being checked.

Description Provides a mechanism for spell-checking text outside of the Words application. You must design a way for passing and returning arrays of misspelled words to and from your application.

To spell check words in a foreign language, use **WP_SET_LANGUAGE@** to set the language.

Example

CHMOD_ABSOLUTE@

Sets a file's permissions

Format CHMOD_ABSOLUTE@(fileOrDir, value)

Arguments

fileOrDr	The file or directory whose access permissions are being changed.
value	The value to which a file or directory is set.

Description Changes the permissions for a file or directory. Only absolute specifications are allowed. For example, you can specify a value of 666. You cannot specify a value of +w.

CLIPART_DIR@

Returns the name of the clipart directory

Format directory = CLIPART_DIR@( )

Description Returns the name of the directory in which clipart is stored. That is, this the directory in which the sample clipart resides.

CLIPBOARD_GET@

Returns the clipboard's contents

Format stringArray = CLIPBOARD_GET@([docTypeCode] [, clipboardName])

Arguments

docTypeCode — The document type code. See **Document Type Codes** for more information. If you omit this argument, the data is assumed to be an ASCII ELF array.

clipboardName — One of the following names:

- PRIMARY
- CLIPBOARD

Description Returns clipboard data as an array of information.

See also **CLIPBOARD_PUT@**

CLIPBOARD_PUT@

Places information onto the clipboard

Format CLIPBOARD_PUT@(data[, docTypeCode[, clipboardName]])

Arguments

data — The information being placed on the clipboard.

docTypeCode — The document type code. See **Document Type Codes** for more information. If you omit this argument, the data is assumed to be an ASCII ELF array.

clipboardName — One of the following names:

- PRIMARY
- CLIPBOARD

Description Places information onto the clipboard. To retrieve this information, use **CLIPBOARD_GET@**.

CLIPBOARD_PUT_FILE@

Places information from
a file onto the clipboard

Format CLIPBOARD_PUT_FILE@(filename [, docTypeCode[, clipboardName]])

Arguments filename The name of a file. The contents of this file are placed on the clipboard.

docTypeCode
The document type code. See **Document Type Codes** for more information. If you omit this argument, the data is assumed to be an ASCII ELF array.

clipboardName
One of the following names:

- PRIMARY
- CLIPBOARD

See also **CLIPBOARD_PUT@**, **CLIPBOARD_GET@**

CLOSE_FILE@

Closes an operating system file

Format CLOSE_FILE@(filename)

Arguments filename The full pathname of the file to close.

Description Closes a file opened by **OPEN_ASCII_FILE@**. If an ELF macro that opens and closes files terminates before the files are closed, the files remain open. Place an error handler that closes files in any macro that opens them to avoid files being left open. See also **ERROR@**.

CLOSE_FILE@ clears all outstanding file locks on the target file.

Example

CMYK_TO_RGB@

Converts a color from one system to another

Format CMYK_TO_RGB@(cyan, magenta, yellow, black, red, green, blue)

Arguments

cyan	The color's cyan component.
magenta	The color's magenta component.
yellow	The color's yellow component.
black	The color's black component.
red	The color's red component.
green	The color's green component.
blue	The color's blue component.

Description Changes a color from the CMYK system to the RGB system. The three RGB colors are set by this macro. That is, you only specify the four CMYK values.

See also WP_HSB_TO_RGB@
WP_RGB_CMYK@
WP_RGB_HSB@

COLOR_STATE@

Returns the virtual depth of the display

Format depth = COLOR_STATE@()

Description Returns the virtual depth of the display. This is the number of bits used to display information on your display screen. For example, 2 is black and white; 8 is usually grayscale.

COLOR_TEMPLATE_DIR@

Returns the color template directory

Format COLOR_TEMPLATE_DIR@()

Description Returns the directory containing the color handout templates used by Applixware Presents.

CURRENT_WINDOW_NUM@

Returns the small id of the current Applix*ware* window

Format id = CURRENT_WINDOW_NUM@()

Description Returns the current small identification number (a number from 0 to 21).

COMMA_SEPARATE@

Returns an array converted to a string

Format string = COMMA_SEPARATE@(inArray)

Arguments

inArray	A one-dimensional array of values. If inArray is not a one-dimensional array or if any element in inArray is a sub-array, NULL is returned.

Description Returns a string that contains the concatenation of all elements in an array. A comma is inserted into this string to indicate the end of one element and the start of another. For example, a two element array containing the elements one and two is converted to the string one, two.

Example

See also ARRAY_TO_STRING@
COMMA_SPLIT@

COMMA_SPLIT@

Breaks a comma-separated string into an array

Format array = COMMA_SPLIT@(string)

Arguments

string	A comma-separated string.

Description Breaks a comma-separated string into an array.

Example

See also ARRAY_FROM_STRING@
COMMA_SEPARATE@

COMMAND_LINE_OPTIONS@

Returns a string array containing Applix*ware* command line options

Format array = COMMAND_LINE_OPTIONS@()

Description Returns an array of the command line options used when Applix*ware* was last invoked.

Example

COMPOSE_TIME@

Changes a data_time_array_ structure
into a date number

Format COMPOSE_TIME@(format date_time_array_ date_array)

Arguments format date_time_array_ date_array

Description Accepts a date_time_array_ structure and converts it to a date/time value. The date_time_array_ contains the following values:

· Seconds:	0 - 59
· Minutes:	0 -59
· Hour:	0 - 23
· Day:	1 - 31
· Month:	0 - 11
· Year:	(year - 1900)
· Weekday:	Sunday is 0
· Yearday:	0 - 365
· Is daylight savings time:	1 if TRUE, 0 if FALSE

Example

COPY_ARRAY@

Copies the passed array the number of times indicated

Format	newArray = COPY_ARRAY@(array, copies)	
Arguments	array	The array to copy.
	copies	Number of copies to make.

Description Copies the passed array the amount of times indicated. For example, if you are copying a 10-element array and array[1] contains 100, then a call to COPY_ARRAY@(array, 4) creates a new array of 40 elements. Elements 1, 11, 21, and 31 contain the value 100.

COPY_CLIPBOARD_TO_FILE@

Copies clipboard's contents to a file

Format	COPY_CLIPBOARD_TO_FILE@(filename[, docType])	
Arguments	filename	The name of the file into which the contents of the clipboard will be pasted.
	docType	A code signifying the structure of the information within the clipboard. For a list of these data types, see **Document Type Codes**. The only valid codes are those beginning with the string AX_DOC_TYPE_ASTERX.

Description Copies information from the clipboard into filename. The internal format of this information is defined by docType. If you do not specify a docType, the default is string. The only valid codes are those beginning with the string AX_DOC_TYPE_ASTERX.

See also COPY_FILE_TO_CLIPBOARD@

COPY_FILE@

Copies a file

Format COPY_FILE@(filename, destination)

Arguments	filename	The name of the file being copied.
	destination	The path name to which the file will be written.

Description Copies a file from one place in the file system to another. Use the full path name if the file you copy is not in the current directory. The filename and destination arguments must be different.

Example

See also MOVE_FILE@

COPY_FILE_TO_CLIPBOARD@

Copies a file to the clipboard

Format COPY_FILE_TO_CLIPBOARD@(filename, docType)

Arguments	filename	The name of the file being copied to the clipboard.
	docType	A code signifying the structure (or source) of the information. For a list of these data types, see **Document Type Codes**.

See also COPY_CLIPBOARD_TO_FILE@

COS@

Returns the cosine of an angle based on its radian value

Format value = COS@(radianValue)

Arguments	radianValue	The degree of an angle.

Description A radian value must be a number between -1 and 1.

Example

See also LOG@
LOG10@
SIN@
SQR@

CREATE_DIR@

Creates a directory

Format CREATE_DIR@(name)

Arguments

name	The name for the new directory.

Description Creates directory name. If you do not specify a path name, the directory is created in the current directory. If a directory with name already exists, an error is displayed and no directory is created.

Example

See also CURRENT_DIR@

CREATE_GRAPHIC@

Create an instance of the Graphics Editor

Format CREATE_GRAPHIC@(gfx, [programID])

Arguments

gfx	The graphics handle for the newly created graphics editor. This is a returned binary value.
programID	An optional value indicating the task to which the handle is assigned.

Description Creates an internal instance of the Graphics Editor. A *handle* to this internal editor is returned. After an internal instance is created, you can directly manipulate graphic objects within ELF without incurring the overhead of the Graphics Editor (which is designed for interactive use). That is, you will use this handle as an argument to many graphic functions.

See also ASSIGN_GRAPHIC@
DESTROY_GRAPHIC@
GET_GRAPHIC@
LOAD_GRAPHIC@
READ_GRAPHIC_FILE@

CREATE_TEMP_FILE@

Creates a temporary file

Format tempFileName = CREATE_TEMP_FILE@([name])

Arguments	name	The prototype file name of the temporary file consisting of the path name, any characters you want to include in the file name, and from six to fourteen percent (%) signs.
		If this parameter is omitted, a default prototype pattern of "%%%%%%%%".

Description Creates a temporary file with a name based on the prototype name you provided and returns the name of the file. The file has a zero file length and permissions set to:

rw------

All temporary files are automatically deleted when the task in which they were created ends. When Applix*ware* terminates, all temporary files are deleted. Use **DELETE_FILE@** to delete a temporary file before a task is completed.

CURRENT_DIR@

Returns the current directory name

Format directory = CURRENT_DIR@()

Description Returns a string indicating the full path name of the current directory.

Example

CURRENT_TIME@

Returns the current time

Arguments time = CURRENT_TIME@()

Description Returns the current system time, based on the number of seconds elapsed since January 1, 1970 GMT. Use **DATE_FORMAT@** to change the value returned by CURRENT_TIME@ into a readable date and time.

Example

CURRENT_WINDOW_NUM@

Returns the current window's identification number

Format colNum = CURRENT_WINDOW_NUM@()

Description The window's identification number is a value between 0 and 20 that is assigned when the window is created. -1 is returned if there is no active window. If the window is busy (in an hourglass state), the macro waits until the action is completed before a value is assigned.

Example

See also CURRENT_WINDOW_TITLE@
WINDOW_INFO@
SELECT_WINDOW@
TASK_ID@

CURRENT_WINDOW_TITLE@

Returns the currently active window's title

Format title = CURRENT_WINDOW_TITLE@()

Description Returns a string indicating the title of the current window.

Example

See also CURRENT_WINDOW_NUM@
SELECT_WINDOW@
WINDOW_INFO@

CUSTOM_DIR@

Returns the name of the Applix*ware* custom directory

Format dir = CUSTOM_DIR@()

CUSTOM_LANG_DIR@

Returns directory containing *axlocal* language-dependent files

Format dir = CUSTOM_LANG_DIR@()

Description Returns the name of the directory in which the axlocal language-dependent files are located. For example, this macro could return *install_dir*/lang.

DATA_APPLICATION_DLG@

Invokes Applixware Data

Format DATA_APPLICATION_DLG@([filename[, menubarID[, windowlessFlag[, title]]]])

Arguments

filename	The name of the file that will be loaded when the window is opened. (This argument is optional.)
menubarID	The number of a menu bar to be associated with this window. (This argument is optional.) This number should be between 200 and 299.
windowlessFlag	A Boolean value where TRUE indicates that no window will be displayed. FALSE is the default.
title	An optional title for the window. This value will replace the file name that usually appears at the top of the window.

Description Invokes Applixware Data. Using this macro, you can optionally:

- Load a file into the Data window.

- Replace the default menu bar with one of your choosing.
- Run Data in the background.
- Give the Data window a new title.

DATA_TYPE@

Returns the data type of the passed element

Format dataType = DATA_TYPE@(datum)

Arguments datum The data type that is returned.

Description Returns the data type of the passed element. datum is one of the following:

0	Type not yet determined
1	number
2	text
5	built-in function
6	recordable C call
7	function key descriptor
8	array
9	variable data type
10	argument data type
11	global variable
12	extern variable
13	non-recorded function data type
14	recordable function
15	non-recordable top-level function
16	remove function data type
17	functions invokable by command key
18	label data type
19	integer data type
21	object
22	binary data
23	write method
24	read method
25	object local data variable

DATE@

Converts a complete year to a serial number

Format DATE@(year, month, day)

Arguments

year	A whole number between 0 and 99.
month	A whole number between 1 and 12.
day	A whole number between 1 and 31.

Description The DATE@ function converts a complete date (year, month, day) into a serial number. For example:

DATE@(84,8,6)	returns 30900
DATE@(83,7,9)	returns 30506
DATE@(85,3,9)	returns 31115

Applix*ware* checks the number you enter to make sure it is a valid entry. For example, if you enter DATE@(84,22,3), your cell displays ERROR because there is no 22nd month.

The DATE@ function is most commonly used to perform arithmetic operations involving date values. The formula DATE@(84,8,6)-DATE@(83,7,9) calculates the number of elapsed days between July 9, 1983 and August 6, 1984. It returns the value 394.

DATEVALUE@

Returns the number of days since 12/31/1899

Format DATEVALUE@(dateString)

Arguments

dateString	A string containing a formatted date.

Description DATEVALUE@ returns a number representing the dateString argument. The number returned is the total number of days since 12/31/1899. For example, DATEVALUE@("1/1/1900") is 1, DATEVALUE@("1/2/1900") is 2, and so on.

The dateString is a formatted date. For a list of valid date formats, choose Style Ý Numbers from Applixware Spreadsheets and use one of the formats displayed in the scrolling list when you click on Date.

If the date format is ambiguous, such as "10/07/95" which could mean either October 7 or July 10, the first date format in the Number Style list is used.

Time information in dateString is ignored.

DATESTR@

Returns a formatted date string

Format DATESTR@(dateNumber, format, fullYearFlag)

Arguments

dateNumber — A date value as returned by the macro DATEVALUE@. This is the number of days after 12/31/1899 that the given date falls.

format — a number from 1 to 20 indicating the format of the returned string.

For-mat Value	Result
1	May 18, 1996
2	May 18, 1996
3	18 May 1996
4	05/18/1996
5	18.05.1996
6	1996-05-18
7	1996-05-18
8	1996 05 18
9	1996 05 18
10	19960518
11	19960518
12	18/05/1996
13	18.05.1996
14	May 18,1996
15	May 1996
16	May 1996
17	May 1996
18	05/18
19	1996 05
20	1996 05

fullYearFlag — if TRUE, display the year as a four digit year. For example, 05/18/87 is displayed as 05/18/1987 if the fullYearFlag is TRUE.

Description Returns a formatted date string. The format of the string is determined by the format variable. The year can be in either a two-digit (87) or four digit (1987) format.

DATETIME_CHANGE_FORMAT@

Changes a formatted time/date string to another format

Format DATETIME_CHANGE_FORMAT@(dtString, dtFormat)

Arguments

dtString	A formatted datetime string
dtFormat	A string or number indicating the new format for the datetime information.

Description Converts a formatted datetime string into a datetime string with a different format. The dtFormat argument can be either a string, or a number, as defined in the file datetime_.sp.

The example below, retrieves the current date and time, and displays the information in two different formats.

```
macro test
var x, y

x = datetime_now@()
info_message@("The date: " ++   DATETIME_TO_STRING@(x, 1))
' Prints 'The date: August 9, 1996'
y = DATETIME_CHANGE_FORMAT@(x, "Mm d, yyyy")
' Prints 'The date: Aug 9, 1996'
info_message@("The date: " ++ y)
endmacro
```

See also Date Formats lists the numbers and strings that are used in the dtFormat argument.

DATETIME_NOW@

Return the current date and time in an array

Format DATETIME_NOW@()

Description Returns an array containing a datetime_ format. This format is defined in the file datetim_.am. Seconds are rounded to 4 second intervals.

DATETIME_TO_STRING@

Converts a datetime_ array to a string

Format DATETIME_TO_STRING@(format datetime_ dtArray, dtFormat)

Arguments

dtArray	A datetim_ array, as defined in **datetim_.am**.
dtFormat	A string or number indicating the format for the datetime information.

Description Converts a datetim_ array into a string. The dtFormat argument can be either a string, or a number, as defined in the file datetime_.sp. Several examples follow:

```
DATETIME_TO_STRING@(array, 1)          ' August 9, 1996
DATETIME_TO_STRING@(array, "Mmmm d, yyyy")
                                   ' August 9, 1996

DATETIME_TO_STRING@(array, 2)          ' Aug 9, 1996
DATETIME_TO_STRING@(array, "Mm d, yyyy")
                                   ' Aug 9, 1996

DATETIME_TO_STRING@(array, 4)          ' 08/09/96
DATETIME_TO_STRING@(array, "mm/dd/yy")
                                   ' 08/09/96
```

See also **Date Formats** lists the numbers and strings that are used in the dtFormat argument.

DATESTR@

DATETIME_STRING_TO_DATETIME@

Converts a formatted
string to a datetime_ array

Format DATETIME_STRING_TO_DATETIME@(dtString)

Arguments

dtString	A string or number indicating the format for the datetime information.

Description Converts a formatted string into a datetim_ array. The datetim_ array is defined in the ELF include file datetime_.am.

See also Date Formats lists the possible formats for the dtString argument.

DATETIME_STRING_TO_TIME_VALUE@

Returns the UNIX time

Format DATETIME_STRING_TO_TIME_VALUE@(dtString)

Arguments

dtString	A formatted datetime string

Description Returns (18000 + the number of seconds since January 1, 1970). This macro is used to fill the EDAT, DDAT, RDAT fields in the OM_SEND_MESSAGE@ macro.

See also OM_SEND_MESSAGE@

DAYS360@

Returns the number of days between two dates

Format DAYS360@(start_date, end_date, method)

Arguments

start_date	Start date of the range of days to be measured.
end_date	End date of the range of days to be measured.
method	A boolean. If TRUE, use the US method to measure the number of days. If FALSE, use the European method.

Description DAYS360@ returns the number of days between two dates based on a 360-day year (twelve 30-day months). This artificial 360-day year is routinely used in the securities industry.

The arguments start_date and end_date can be either text strings that represent the month, day, and year, such as "10/24/95", or serial numbers representing the dates.

The method is a logical argument specifying whether to use the U.S. or European method to calculate the function.

Method	Definition
FALSE or omitted	US (NASD). If the starting date is the 31st of a month, it becomes equal to the 30th of the same month. If the ending date is the 31st of a month and the starting date is less than the 30th of a month, the ending date becomes equal to the 1st of the next month; otherwise the ending date becomes equal to the 30th of the same month.
TRUE	European method. Starting dates or ending dates that occur on the 31st of a month become equal to the 30th of the same month.

For example, DAYS360@("1/30/95", "2/1/95") equals 1.

DATE_FORMAT@

Converts a UNIX time/date value to a time/date string

Format string = DATE_FORMAT@(date, format)

Arguments	date	The UNIX date/time value, indicating the number of seconds elapsed since January 1, 1970, GMT. Use CURRENT_TIME@ to get this value.
	format	A number or combination of numbers indicating how the date/time string should be represented.

Description Returns a string version of the current date and/or time. Daylight savings time is considered. See also **CURRENT_TIME@**. The following is a list of the date/time formats. You can combine elements from the table together to form a unified date/day/time format. For example, a valid format might be the number 4202.

NOTE: The following list approximates what you may receive. The **datetime.sp** file contains a description of exactly what will be returned.

Date Formats

#	Format	Description
1.	Mmmm d, yyyy	December 2, 1992 or December 12, 1992
2.	Mm d, yyyy	Dec 2, 1992 or Dec 12, 1992
3.	d Mm yyyy	2 December 1992 or 12 December 1992
4.	mm/dd/yy	12/02/92
5.	dd.mm.yy	02.12.92

6. yyyy-mm-dd	1992-12-02
7. yy-mm-dd	92-12-02
8. yyyy mm dd	1992 12 02
9. yy mm dd	92 12 02
10. yyyymmdd	19921202
11. yymmdd	921202
12. dd/mm/yy	02/12/92
13. dd.mm.yyyy	02.12.92
14. Mm dd, yyyy	Dec 02, 1992 (the Inbox date format)
15. Mmmm yyyy	December 1992
16. Mm yyyy	Dec 1992
17. Mm yy	Dec 92
18. mm/dd	12/02
19. yy mm	92/12
20. yyyy mm	1992/12

Time formats:

#	Format	Description	
100	hh:mi pm	09:34 pm	12-hour format
200	hh:mipm	09:34pm	12-hour format
300	HH:mi	21:34	24-hour format
400	HH.mi	21.34	24-hour format
500	HHmi	2134	24-hour format
600	HH:mi:ss	21:34:05	24-hour format

Day of the week formats:

0	No day of week
1000	Tue <date/time>
2000	<date/time> Tue
3000	Tuesday <date/time>
4000	<date/time> Tuesday

Example

DATE_LAST_MODIFIED@

Returns the time a file was last modified

Format time = DATE_LAST_MODIFIED@(name)

Arguments name The file's relative or absolute path name.

Description Returns a value representing the time a file was last modified. This time is based on the number of seconds elapsed since January 1, 1970 GMT. Returns the value 0 if the specified file does not exist.

DAY@

Extracts the day of the month from a serial date number

Format DAY@(dateNumber)

Arguments dateNumber A serial date number.

Description Extracts the day of the month (1-31) from a serial date number. You can enter a serial date number as an argument for the DAY@ function. A formula that contains the serial number 30899 (August 6, 1984) returns 6.

You can also use the **DATE@** or **TODAY@** function as an argument in the DAY@ function. For example, the formula DAY@(DATE@(84,1,3)) returns 3.

DB_ACCEPT_POKES@

Indicates which messages a dialog box macro will accept

Format DB_ACCEPT_POKES@(dbox, codeArray)

Arguments

dbox	The name of the dialog box variable.
codeArray	An array of poke message codes that will be accepted by the dialog box macro.

Description Used with DB_SEND_POKE@ and DB_GET_POKE@ to send messages between a task and a dialog box task. A dialog box macro will only respond to messages having poke codes assigned to it using DB_ACCEPT_POKES@.

See also **DB_GET_POKE@**

DB_SEND_POKE@

DB_CANCELLED@

Indicates whether a dialog box was cancelled

Format flag = DB_CANCELLED@(dbox)

Arguments	dbox	The name of the dialog box variable.

Description Returns TRUE if a user presses CANCEL or hits ESCAPE in a dialog box; otherwise it returns FALSE. Push buttons of type Dismiss cancel a dialog box when pressed.

DB_CLOSE@

Closes a dialog box

Format DB_CLOSE@()

Description Closes (removes) a dialog box. Execute & Dismiss and Dismiss push buttons automatically call DB_CLOSE@. Using DB_CLOSE@ to close a dialog box is only necessary when you want a dialog box to close without the use of Execute & Dismiss or Dismiss buttons.

See also DB_CANCELLED@

DB_CREATE_CTRL@

Creates a dialog box control or label

Format DB_CREATE_CTRL@(dbox, type, name, title, xpos, ypos, default)

Arguments	dbox	The name of the dialog box variable.
	type	One of the following control types: 0 radio button group 1 toggle button 2 option button 3 push button 4 entry box 5 label 6 bitmap (decoration) 7 list box 8 panel/line (decoration) 9 edit box (scrollable multi-line, word-wrapping, text-editing, control). 11 scale 12 table 13 row/col 14 canvas
	name	The string name of the control.

title	A string indicating the control title displayed with the control for push buttons, entry boxes, toggle buttons, radio groups, option buttons, and labels. For bitmaps, title should be the bitmap file name, without the .im extension. The bitmap file must be in your macros directory.
xpos	A number indicating the x-axis position, in pixels, for the top left corner of the control. This position is relative to the top left corner of the dialog box.
ypos	A number indicating the y-axis position, in pixels, for the top left corner of the control. This position is relative to the top left corner of the dialog box.
default	The default selection or text, which depends on the control type you specify:

Entry boxes	A string indicating the text in the entry box. An empty string indicates an empty entry box.
Radio button groups and Option Buttons	A number indicating the radio button, option, or list box item that is selected. Items are numbered from 0. A value of -1 indicates that no item is selected.
List boxes	A string indicating the item that is selected in a list box. If the list box allows multiple selections, you can specify a string array to indicate the items that are selected.
Edit boxes	An array of strings representing the contents. If the string is only one line, it must be an array. That is, it assigns the string to array [0].

Push buttons controls, bitmaps, and panel/line decorations do not have values.

Description Dialog box controls are normally created using the Dialog Box Editor rather than using DB_CREATE_CTRL@.

If you are creating a radio button group, an option button, or a list box, define the items to be used with these controls using DB_CTRL_STRINGS@.

If you are creating an entry box, use DB_CTRL_WIDTH@ to specify the width of the entry box and DB_CTRL_LENGTH@ to indicate how many characters can be typed in the entry box.

If you are creating a list box or an edit box, use DB_CTRL_HEIGHT@ to specify the height, in number of lines, of the list box and DB_CTRL_WIDTH@ to indicate how many characters wide to make the list

If you are creating a panel/line decoration use the following:

DB_CTRL_HEIGHT@
    Indicates the number of pixels high to make the panel/line.

DB_CTRL_WIDTH@
Indicates the number of pixels wide to make the panel/line.

DB_CTRL_LINE_THICKNESS@
Indicate the thickness of the lines in the panel/line decoration.

If you are creating a push button, you will need to use DB_CTRL_BUTTON_TYPE@ to specify the type of push button. For example, you need to indicate if the push button is Execute & Dismiss, Dismiss, Execute, Help, or Bitmap. For more information, see DB_CTRL_BUTTON_TYPE@.

Example

See also DB_CTRL_STRINGS@
DB_CTRL_WIDTH@
DB_CTRL_LENGTH@
DB_CTRL_HEIGHT@
DB_CTRL_LINE_THICKNESS@
DB_DESTROY_CTRL@

DIALOG_WINDOW_ID@

Returns the dialog handle of a window

Format DIALOG_WINDOW_ID@(name)

Arguments	name	The title of the dialog for which you want to find the id number.

Description Returns the dialog handle of the window whose name is name.

DB_CREATE_DIALOG@

Creates a dialog box

Format DB_CREATE_DIALOG@(dboxID, title, width, height)

Arguments	dboxID	A unique identifier, expressed as a number.
	title	A string indicating the title that is displayed in the dialog box.

width	A number indicating the width, in pixels, of the dialog box.
height	A number indicating the height, in pixels, of the dialog box.

Description Creates a new dialog box. Normally, dialog boxes are created using the Dialog Box Editor rather than using DB_CREATE_DIALOG@.

DB_CTRL_ACTIVE_RETURN@

Suspends dialog box operation when an entry box control becomes active

Format DB_CTRL_ACTIVE_RETURN@(dbox, ID, flag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of an entry box that will return control to the dialog box macro when the user moves the cursor into it.
flag	Indicates whether execution of your dialog box macro continues. TRUE indicates that execution continues. The default is FALSE, which suspends execution.

Description Suspends dialog box operation when the cursor is moved into an entry box. This action differs from using DB_CTRL_TYPING_RETURN@, which suspends operation when a character is typed into the entry box.

See also DB_CTRL_RETURN_ON_CHANGE@
DB_CTRL_TYPING_RETURN@

DB_CTRL_ALTERNATE_PIXMAP@

Specifies an alternate array of pixels

Format DB_CTRL_ALTERNATE_PIXMAP@(dbox, ID, filename)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
filename	The name of the file containing a pixmap.

Description Specifies an alternate array of pixels (pixmap) used for drawing the static choice for row/column pull down controls. That is, this is the pattern displayed when the user chooses one of the pixmaps in a row/column pull down.

If you do not use this macro, the pixmap chosen in the row/column display is displayed.

See also DB_CTRL_PIXMAPS@

DB_CTRL_ASSIGN_GRAPHIC@

Assign the graphic to control

Format DB_CTRL_ASSIGN_GRAPHIC@(dbox, ID, gfx, format gr_inset@ inset)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
gfx	The graphics handle.
inset	Information defining the graphic.

Description Assigns a graphic to a dialog box control. After a graphic is associated with a dialog box, it will be drawn whenever the dialog box is displayed

The definition of the gr_inset@ format is as follows:

```
format gr_inset@
        clear_mode,           '"none", "widget", "window"
        scale_mode,           `"clip to fit", "scale to fit", scale if clips"
        proportional,         'Boolean
        prescale_x,           '100 is 1X
        prescale_y,           '100 is 1X
        format gr_area@ src,        'source position and size in current graphic units
        format gr_area@ clip        'destination position and size in current graphic units
        display_mode,         'special display effects
        toggle_wait_cursor,   'should the inset display a wait cursor
        page_numer,           'which page in the graphic to inset
        widget                'widget to draw onto
```

See also DB_CTRL_RENDER_GRAPHIC@

DB_CTRL_BUTTON_TYPE@

Specifies the type of a push button in a dialog box

Format DB_CTRL_BUTTON_TYPE@(dbox, ID, value)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

value	Indicates one of the following types for the push button control:

0 = Normal	Normal push buttons have none of the special attributes provided with Execute & Dismiss, Dismiss, Execute, Help, and Bitmap buttons.
1 = Execute & Dismiss	Execute & Dismiss are usually designated as the default push buttons.
2 = Dismiss	Dismiss push buttons can be selected by pressing the ESC or CANCEL keys.
3 = Help	Help push buttons display a help dialog box or the help window.
4 = Bitmap	Bitmap push buttons do not contain any labels. They are placed under bitmap images so that clicking on the image selects the button.
5 = Execute	When Execute push buttons are pressed, dialog box information is accepted, but the dialog box is not exited.

Description Sets or changes the button type of a push button. Normally, the push button type is specified using the Dialog Box Editor.

See also DB_CREATE_CTRL@

DB_CTRL_BUTTON_UP_SCROLL@

Sets scrolling behavior

Format DB_CTRL__BUTTON_UP_SCROLL@(dbox, ID, flag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
flag	A Boolean value where TRUE indicates that synchronized scrolling will not occur.

Description Sets the scrolling behavior when the user drags the scroll elevator. When set to TRUE, the values in the dialog box do not change until the user releases the button. (That is, scrolling does not occur until a button-up event occurs.) When set to FALSE, scrolling occurs as the elevator is moved within the scrollbar.

DB_CTRL_CALLBACKS@

Associates a callback function with a control

Format DB_CTRL_CALLBACKS@(dbox, ID, cbinfo)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
cbinfo	An array that takes the following form:

```
<<eventName1><funcName1>>[,
<<eventName2><funcName2>>[,
<<eventName3><funcName3>> ] ]
```

The following values may be used for events:

1	Control changed
2	Entry box activated
3	Entry box typing
7	Command Button clicked

For example:

```
cbinfo[0,0] = 1
cbinfo[0,1] = "toggle1_clicked"
```

Description Associates a callback function with a control. That is, when the control is changed (or in the case of an entry box, when the box is activated or when someone types in the entry field), the function associated with the control is invoked.

See also DB_DIALOG_CALLBACKS@

DB_CTRL_CLIENT_DATA@

Attaches data to a control

Format DB_CTRL_CLIENT_DATA@(dbox, ID, value)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
value	An ELF data that you wish to associate with a control. This value can be a format or an array.

Description	Associates the contents of value with any dialog box control. This data is not used or displayed by the dialog box. Instead, it is stored with the dialog box for later use by your program.
	Use **DB_CTRL_GET_CLIENT_DATA@** to retrieve this information.

DB_CTRL_COLOR@

Sets the foreground color for row/column controls

Format DB_CTRL_COLOR@(dbox, ID, valueArray)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
valueArray	The color information as described below

Description Sets the foreground color for a row/column control. The structure of the color information is as follows:

colorSpace	One of the following: WCSPACE#RGB 1 WCSPACE#HSB 5
colorValueArray	If the color space is RGB, a sub-arrary of three values corresponding to the red, green, and blue components of the color. If the color space is HSB, a sub-array of three values corresponding to the hue, saturation, and brightness components of the color.
name	A *nickname* for the color; for example, "magenta" or "red".

Note Do not use this macro for setting colors for other controls. Instead, use one of the following controls:

· **DB_CTRL_LABEL_COLOR@**

· **DB_CTRL_TEXT_COLOR@**

· **DB_CTRL_WIDGET_COLOR@**

See also **DB_CTRL_GET_COLOR@**

DB_CTRL_COLUMNS@

Sets the number of columns in a row and column control

Format DB_CTRL_COLUMNS@(dbox, ID, numCols)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
numCols	The number of columns in a row and column pull down.

Description Determines the number of rows and columns used in a row and column pull down.

For example, suppose there are 12 pixmaps and you have four columns. The result is that you have created a pull down with three rows and four columns.

Another example is in the Equation Editor's icon pull down lists.

DB_COMBO_EDITABLE@

Makes a combo box editable

Format DB_COMBO_EDITABLE@(dbox, ID, editFlag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of a combo box control.
editFlag	Set to TRUE to make the combo box editable

Description Sets the editability of a combo box. If editFlag is TRUE, the combo box becomes editable. If editFlag is FALSE, the combo box cannot be edited.

See also DB_COMBO_GET_EDITABLE@

DB_COMBO_GET_EDITABLE@

Returns TRUE if the combo box is editable

Format DB_COMBO_GET_EDITABLE@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of a combo box control.

Description Returns TRUE if the combo box is editable. Otherwise, this macro returns FALSE.

See also **DB_COMBO_EDITABLE@**

DB_CTRL_COPY@

Copies selected text from an edit box to the clipboard

Format DB_CTRL_COPY@(dbox)

Arguments	dbox	The name of the dialog box variable.

Description Copies selected text from an edit box and places the text in the clipboard.

See also **DB_CTRL_CUT@**
DB_CTRL_PASTE@
DB_CREATE_CTRL@

DB_CTRL_CUT@

Cuts selected text from an edit box to the clipboard

Format DB_CTRL_CUT@(dbox)

Arguments	dbox	The name of the dialog box variable.

Description Cuts selected text from an edit box and places the text in the clipboard.

See also **DB_CTRL_COPY@**
DB_CTRL_PASTE@

DB_CTRL_DEFAULT_BUTTON@

Sets the default push button

Format DB_CTRL_DEFAULT_BUTTON@(dbox, ID, flag)

Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the control.

	flag	Indicates whether to make the specified push button the default button. If TRUE, the push button is the default button. The default is FALSE.

Description Only one push button in a dialog box can be the default push button. If you make a button the default button using DB_CTRL_DEFAULT_BUTTON@, the button that was previously the default automatically loses its default state.

Specifying flag as FALSE is useful when you don't want any push buttons in a dialog box to be a default button.

Example

DB_CTRL_DISPLAY@

Hides or displays a dialog box control

Format DB_CTRL_DISPLAY@(dbox, ID, flag)

Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the control.
	flag	TRUE causes the control to be displayed. FALSE causes the control to be hidden. The default is TRUE.

Description Hides or displays a control. Hidden controls are used when the display of a control may only be appropriate based on the value of another control in the dialog box.

See also DB_CTRL_GET_DISPLAY@

DB_CTRL_EDITMASK@

Sets an edit mask for an entry field

Format DB_CTRL_EDITMASK@(dbox, ID, mask)

Method this.edit_mask@(mask)

Arguments	dbox	The name of the dialog box variable.
	ID	The string name of and entry field control.
	mask	A combination of placeholder and literals, according to the description the follows.

Description Establishes an edit mask for restricting keystroke input in the edit field.

For example:

```
mask = "(999) 999-9999"
DB_CTRL_EDITMASK@(dbox, "Entry Box", mask)
```

This would establish an entry field for entering phone numbers.

An edit mask should contain just two types of characters: *placeholders*, and *literals*. A masked entry field accepts input only at the placeholder positions. Each placeholder accepts a particular set of input characters.

There are currently seven placeholders defined: 9#A?&!~

9	accepts the digits 0-9
&	accepts any printable character
#	accepts 0-9 plus - + .
!	accepts a-Z A-Z, converts to uppercase
A	accepts 0-9 plus A-Z
~	accepts a-Z A-Z, converts to lowercase
?	accepts a-z A-Z

Literals are mask characters which always appear in the entry field, and can't be typed over or deleted. There are 4 predefined literals: $()-

Any other character may be supplied as a literal by preceding it with a backslash within the mask. For example, supplying \@ in the mask results in a literal @ at that position. If a mask character is not preceded by a backslash, and is not among the set of placeholders, it will be treated as a literal space character. Such a literal may, in future Builder releases, conflict with new placeholder definitions, so make use of the backslash when specifying literals other than $()- and space character.

A masked entry field's value in its empty state will appear as a string containing the space character at placeholder positions, and literals elsewhere.

An edit mask also restricts the length of the field's value -- if the mask is 10 characters in length, then the entry field will also be restricted to a length of 10 characters. So, in an extended phone number mask such as: (999) 999-9999 x9999, the field value will always contain 20 characters; if the extension is only two digits long, then the field will have two trailing space characters.

DB_CTRL_FONT@

Sets the X-font for a widget

Format DB_CTRL_FONT@(dbox, ID, xFont)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
xFont	The X-font being set for the control.

Description Sets the font for a control. Use **LIST_FONT_FAMILIES@** to obtain a list of fonts.

See also **DB_CTRL_GET_FONT@**

DB_CTRL_GET_CLIENT_DATA@

Retrieves data bound to a control

Format value = DB_CTRL_GET_CLIENT_DATA@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Retrieves the value of the data stored with a dialog box. This data was assigned to the control using **DB_CTRL_CLIENT_DATA@**.

DB_CTRL_GET_COLOR@

Returns the foreground color

Format valArray = DB_CTRL_GET_COLOR@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns the foreground color for a row/column control. The structure of the returned information is as follows:

colorSpace	One of the following:

	WCSPACE#RGB	1
	WCSPACE#HSB	5
colorValueArray	If the color space is RGB, a sub-array of three values corresponding to the red, green, and blue components of the color. If the color space is HSB, a sub-array of three values corresponding to the hue, saturation, and brightness components of the color.	
name	A *nickname* for the color; for example, "magenta" or "red".	

DB_CTRL_GET_DISPLAY@

Indicates if a control is visible

Format flag = DB_CTRL_GET_DISPLAY@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns TRUE if the control is visible; otherwise, this macro returns FALSE.

See also **DB_CTRL_DISPLAY@**

DB_CTRL_GET_EDITMASK@

Returns the edit mask for an Entry Field

Format mask = DB_CTRL_GET_EDITMASK@(dbox, ID)

Method mask = this.edit_mask@()

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns the edit mask for an entry field. For more information on edit masks, see the description of the macro DB_CTRL_EDITMASK@.

See also **DB_CTRL_EDITMASK@**

DB_CTRL_GET_FONT@

Returns the X-font name used for a control

Format	font = DB_CTRL_GET_FONT@(dbox, ID)	
Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the control.

Description Returns the X-font name used to display a widget. If a font was not explicitly set (which means that the control is using the default font), this macro returns NULL.

Use DB_XLATE_FONT@ to convert an Applixware font to an X-font.

See also DB_CTRL_FONT@

DB_CTRL_GET_GRAYED@

Returns whether control is dimmed

Format	flag = DB_CTRL_GET_GRAYED@(dbox, ID)	
Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the control.

Description Returns TRUE if the control is dimmed or grayed out.

See also DB_CTRL_GRAYED@

DB_CTRL_GET_INSET@

Displays an inset on a dialog box widget

Format	format ax_inset@ inset = DB_CTRL_GET_INSET@(dbox, wdef)	
Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the control

Description Returns an array of format **ax_inset@ format**, containing information about an inset associated with a control in a dialog box. The ID argument must be the Control ID of a Panel or Canvas control onto which an inset has been rendered.

See also **DB_CTRL_INSET@**

DB_CTRL_GET_HEIGHT@

Returns a control's height

Note This macro is obsolete. Use **DB_CTRL_GET_SIZE@** instead.

Format height = DB_CTRL_GET_HEIGHT@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns the control's height.

See also **DB_CTRL_HEIGHT@**

DB_CTRL_GET_LABEL_COLOR@

Returns a control's color

Format valArray = DB_CTRL_GET_LABEL_COLOR@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns an array of color information that specifies a label's color. The structure of this information is as follows:

colorSpace	One of the following: WCSPACE#RGB 1 WCSPACE#HSB 5
colorValueArray	If the color space is RGB, a sub-array of three values corresponding to the red, green, and blue components of the color. If the color space is HSB, a sub-array of three values corresponding to the hue, saturation, and brightness components of the color.
name	A *nickname* for the color; for example, "magenta" or "red".

See also	DB_CTRL_LABEL_COLOR@
	DB_CTRL_GET_TEXT_COLOR@
	DB_CTRL_GET_WIDGET_COLOR@

DB_CTRL_GET_LABEL_FONT@

Returns the name of the font used for a label

Format fontName = DB_CTRL_GET_LABEL_FONT@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns a string that contains the name of the font used to display a control's label. For example, this macro could return the word "Courier". If you are using the default font, NULL is returned.

See also	DB_CTRL_LABEL_FONT@
	DB_CTRL_GET_LABEL_FONT_SIZE@
	DB_CTRL_GET_LABEL_FONT_SLANT@
	DB_CTRL_GET_LABEL_FONT_WEIGHT@
	DB_CTRL_GET_LABEL_SHADOW@

DB_CTRL_GET_LABEL_FONT_SIZE@

Returns the point size of a label's text

Format pointSize = DB_CTRL_GET_LABEL_FONT_SIZE@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns a number that contains the size of the font used to display a control's label. For example, this macro could return the number 14, which is the default point size. If you have not set a point size (or a font), NULL is returned.

See also	DB_CTRL_LABEL_FONT_SIZE@
	DB_CTRL_GET_LABEL_FONT@
	DB_CTRL_GET_LABEL_FONT_SLANT@

DB_CTRL_GET_LABEL_FONT_WEIGHT@

DB_CTRL_GET_LABEL_SHADOW@

DB_CTRL_GET_LABEL_FONT_SLANT@

Returns the slant value

Format num = DB_CTRL_GET_LABEL_FONT_SLANT@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns a number indicating the "slant" value used to display a label's text. The only meaningful values are 0 (no slant) and 1 (italics). All other values are undefined.

If you are using the default font and have not assigned a slant value, NULL is returned.

See also DB_CTRL_LABEL_FONT_SLANT@

DB_CTRL_GET_LABEL_FONT@

DB_CTRL_GET_LABEL_FONT_SIZE@

DB_CTRL_GET_LABEL_FONT_WEIGHT@

DB_CTRL_GET_LABEL_SHADOW@

DB_CTRL_GET_LABEL_FONT_WEIGHT@

Returns the weight of a label's text

Format boldVal = DB_CTRL_GET_LABEL_FONT_WEIGHT@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns a number indicating the "weight" used to display a label's text. The only meaningful values are 0 (normal display) and 0 (bold). All other values are undefined.

If you are using the default font and have not assigned a weight value, NULL is returned.

See also DB_CTRL_LABEL_FONT_WEIGHT@

DB_CTRL_GET_LABEL_FONT@

DB_CTRL_GET_LABEL_FONT_SIZE@

DB_CTRL_GET_LABEL_FONT_SLANT@
DB_CTRL_GET_LABEL_SHADOW@

DB_CTRL_GET_LABEL_SHADOW@

Returns Boolean indicating if drop shadow printing is used

Format flag = DB_CTRL_GET_LABEL_SHADOW@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns a Boolean (TRUE/FALSE) value indicating if the label's text is drawn using drop shadow lettering is used.

If you have never set the drop shadow property, NULL is returned.

See also DB_CTRL_LABEL_SHADOW@
DB_CTRL_GET_LABEL_FONT@
DB_CTRL_GET_LABEL_FONT_SIZE@
DB_CTRL_GET_LABEL_FONT_SLANT@
DB_CTRL_GET_LABEL_FONT_WEIGHT@

DB_CTRL_GET_OPTIONAL@

Indicates if an entry field is optional

Format flag = DB_CTRL_GET_OPTIONAL@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns TRUE if the entry field named by ID is optional. If the field is required, FALSE is returned.

See also DB_CTRL_OPTIONAL@

DB_CTRL_GET_PIXMAPS@

Returns the pixmaps in a row/column pull down

Format pixmapArray = DB_CTRL_GET_PIXMAPS@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the row/column control.

Description Returns an array that contains the pixmaps that are used in a row/column pull-down.

See also DB_CTRL_PIXMAPS@

DB_CTRL_GET_SHADOW@

Returns Boolean indicating if a drop shadow exists behind button

Format flag = DB_CTRL_GET_SHADOW@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns a Boolean (TRUE/FALSE) value indicating a drop shadow box is drawn behind a button.

If this property has not been set previous, NULL is returned.

See also DB_CTRL_SHADOW@

DB_CTRL_GET_SIZE@

Returns a control's position, width and height

Format sizeArray = DB_CTRL_GET_SIZE@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns a four element array whose contents are as follows:

sizeArray[0]	The *X* coordinate of the control's upper left corner.
sizeArray[1]	The *Y* coordinate of the control's upper left corner.

sizeArray[2]	The control's width.
sizeArray[3]	The control's height.

See also DB_CTRL_HEIGHT@

DB_CTRL_WIDTH@

DB_CTRL_XPOS@

DB_CTRL_YPOS@

DB_CTRL_GET_STRINGS@

Returns the control's string values

Format stringArray = DB_CTRL_GET_STRINGS@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns a string array. In this array, each element is one of the strings labels or items used in a control that can display multiple strings:

List Boxes
    The strings are the items in the list box.

Radio Button Groups
    The strings are the radio button labels.

Option Buttons
    The strings are the options from the option button menu.

See also DB_CTRL_STRINGS@

DB_CTRL_GET_TEXT_COLOR@

Returns the text's foreground color

Format value = DB_CTRL_GET_TEXT_COLOR@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns an array of color information that specifies the text color for a control. The structure of this information is as follows:

colorSpace   One of the following:

	WCSPACE#RGB 1 WCSPACE#HSB 5
colorValueArray	If the color space is RGB, a sub-array of three values corresponding to the red, green, and blue components of the color. If the color space is HSB, a sub-array of three values corresponding to the hue, saturation, and brightness components of the color.
name	A *nickname* for the color; for example, "magenta" or "red".

See also DB_CTRL_TEXT_COLOR@

DB_CTRL_GET_LABEL_COLOR@

DB_CTRL_GET_WIDGET_COLOR@

DB_CTRL_GET_TEXT_FONT@

Returns the name of the font used for text

Format fontName = DB_CTRL_GET_TEXT_FONT@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns a string that contains the name of the font used to display text in an entry field. for example, this macro could return the word "Courier". If you are using the default font, NULL is returned.

See also DB_CTRL_TEXT_FONT@

DB_CTRL_GET_TEXT_FONT_SIZE@

DB_CTRL_GET_TEXT_FONT_SLANT@

DB_CTRL_GET_TEXT_FONT_WEIGHT@

DB_CTRL_GET_TEXT_SHADOW@

DB_CTRL_GET_TEXT_FONT_SIZE@

Returns the point size of an entry field's text

Format pointSize = DB_CTRL_GET_TEXT_FONT_SIZE@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.

ID	The string name of the control.

Description Returns a number that contains the size of the font used to display the text within an entry field box. For example, this macro could return 14, which is the default point size. If you have not set a point size (or a font), NULL is returned.

See also DB_CTRL_TEXT_FONT_SIZE@

DB_CTRL_GET_TEXT_FONT@

DB_CTRL_GET_TEXT_FONT_SLANT@

DB_CTRL_GET_TEXT_FONT_WEIGHT@

DB_CTRL_GET_TEXT_SHADOW@

DB_CTRL_GET_TEXT_FONT_SLANT@

Returns the slant value of an entry field's text

Format slantVal = DB_CTRL_GET_TEXT_FONT_SLANT@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns a number indicating the "slant" value used to display the text within an entry field box. The only meaningful values are 0 (no slant) and 1 (italics). All other values are undefined.

See also DB_CTRL_TEXT_FONT_SLANT@

DB_CTRL_GET_TEXT_FONT@

DB_CTRL_GET_TEXT_FONT_SIZE@

DB_CTRL_GET_TEXT_FONT_WEIGHT@

DB_CTRL_GET_TEXT_SHADOW@

DB_CTRL_GET_TEXT_FONT_WEIGHT@

Returns the weight of an entry field's text

Format weight = DB_CTRL_GET_TEXT_FONT_WEIGHT@ (dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns a number indicating the "weight" used to display the text within an entry field box. The only meaningful values are 0 (normal weight) and 1 (bold). All other values are undefined.

If you are using the default font and have not assigned a weight value, NULL is returned.

See also DB_CTRL_TEXT_FONT_WEIGHT@

DB_CTRL_GET_TEXT_FONT@

DB_CTRL_GET_TEXT_FONT_SIZE@

DB_CTRL_GET_TEXT_FONT_SLANT@

DB_CTRL_GET_TEXT_SHADOW@

DB_CTRL_GET_TEXT_SHADOW@

Returns Boolean indicating if drop shadow is used for entry box text

Format flag = DB_CTRL_GET_TEXT_SHADOW@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns a Boolean (TRUE/FALSE) value indicating if the text in an entry field box is drawn using drop shadow lettering.

If you have never set the drop shadow property, NULL is returned.

See also DB_CTRL_TEXT_SHADOW@

DB_CTRL_GET_TEXT_FONT@

DB_CTRL_GET_TEXT_FONT_SIZE@

DB_CTRL_GET_TEXT_FONT_SLANT@

DB_CTRL_GET_TEXT_FONT_WEIGHT@

DB_CTRL_GET_TITLE@

Returns a control's title

Format title = DB_CTRL_GET_TITLE@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.

	ID	The string name of the control.

Description Returns the title associated with a control. (Not all controls have or use titles.)

See also DB_CTRL_TITLE@

DB_CTRL_GET_TYPE@

Returns a control's type

Format num = DB_CTRL_GET_TYPE@(dbox, ID)

Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the control.

Description Returns the type of control, which is one of the following values:

0	Radio button group
1	Toggle button
2	Option button
3	Push button
4	Entry box
5	Label
6	Bitmap (decoration)
7	List box
8	Panel/line (decoration)
9	Edit box (scrollable multi-line, word-wrapping, text-editing, control).
11	Scale
12	Table
13	Row/column
14	Canvas

DB_CTRL_GET_VALUE@

Returns a control's value

Format value = DB_CTRL_GET_VALUE@(dbox, ID)

Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the control.

Description Obtains the value of a control so you can perform operations based on the control values. Push buttons do not have values. The value returned by DB_CTRL_GET_-VALUE@ depends on the following:

Toggle buttons	TRUE indicates that the toggle button is on. FALSE indicates that the toggle button is off.
Entry boxes	The value returned by an entry box is a string indicating the text in the entry box. An empty string indicates an empty string box.
Scale	The value returned by a scale is a number indicating the current position of the scale.
Radio button groups, Options buttons, List boxes	The value returned by a radio button group, option button, or list box is a number indicating the option that is selected. Options are numbered from 0. If a list box allows multiple selections, the value returned is an array in which each array element corresponds to a list box selection. NULL indicates that no option is selected.

Example

See also DB_CTRL_VALUE@

DB_CTRL_GET_WIDGET_COLOR@

Returns a control's foreground color

Format colorVal = DB_CTRL_GET_WIDGET_COLOR@ (dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns an array of color information that specifies the color for a control. The structure of this information is as follows:

colorSpace	One of the following: WCSPACE#RGB 1 WCSPACE#HSB 5
colorValueArray	If the color space is RGB, a sub-array of three values corresponding to the red, green, and blue components of the color. If the color space is HSB, a sub-array of three values corresponding to the hue, saturation, and brightness components of the color.

name	A *nickname* for the color; for example, "magenta" or "red".

See also DB_CTRL_WIDGET_COLOR@
DB_CTRL_GET_LABEL_COLOR@
DB_CTRL_GET_TEXT_COLOR@

DB_CTRL_GET_WIDTH@

Returns a control's width

Note This macro is obsolete. Use DB_CTRL_GET_SIZE@ instead.

Format width = DB_CTRL_GET_WIDTH@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns the width of a control.

See also DB_CTRL_WIDTH@

DB_CTRL_GET_XPOS@

Returns a control's X-position

Format xPos = DB_CTRL_GET_XPOS@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.

Description Returns the X-position of a control relative to the upper left corner of the dialog box.

See also DB_CTRL_GET_YPOS@
DB_CTRL_XPOS@

DB_CTRL_GET_YPOS@

Returns a control's Y-position

Format yPos = DB_CTRL_GET_YPOS@(dbox, ID)

Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the control.

Description Returns the Y-position of a control relative to the upper left corner of the dialog box.

See also DB_CTRL_GET_XPOS@
DB_CTRL_YPOS@

DB_CTRL_GRAYED@

Grays a dialog box control so that it cannot be chosen

Format DB_CTRL_GRAYED@(dbox, ID, flag)

Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the control.
	flag	If value is TRUE, the control is grayed. If value is FALSE, the control is not grayed. The default is FALSE.

Description Makes a control gray or makes a grayed control normal (not gray). While a control is grayed, it cannot be chosen by the user. Grayed controls are useful when the selection of a dialog box control may not be appropriate for the current state of an application.

Example

See also DB_CTRL_GET_GRAYED@

DB_CTRL_HEIGHT@

Sets a control's height

Format DB_CTRL_HEIGHT@(dbox, ID, height)

Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the control.
	height	For a list box, height is a number indicating the height, in number of lines, to which to set the list box. For edit boxes and panel/line decoration, height is the height of the panel, in pixels.

Description	Sets or changes a control's height. (Normally, the height of list boxes and panel/line decorations is set using the Dialog Box Editor.)
See also	DB_CREATE_CTRL@ DB_CTRL_GET_HEIGHT@ DB_CTRL_WIDTH@

DB_CTRL_HELP@

Assigns a help topic name to a dialog box widget

Format DB_CTRL_HELP@(dbox, ID, topicID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
topicID	The help topic associated with the control

Description Associates a help topic name with a dialog box widget.

DB_CTRL_HORIZ_SCROLL@

Controls horizontal scrollbar display for a list box

Format DB_CTRL_HORIZ_SCROLL@(dbox, ID, flag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
flag	Indicates whether a horizontal scroll bar should be attached to a list box. If flag is TRUE, a horizontal scroll bar is displayed with the list box. The default is FALSE (do not display).

Description Provides a horizontal scroll bar at the bottom of a list box. This space-conserving scroll bar lets you scroll individual entries that exceed the display width of a list box.

You can find horizontal scroll bars at the bottom of a Words, Graphics, Macro, Data, or Spreadsheet window.

Any control initialization statements must appear after DB_LOAD@ and before DB_DISPLAY@ in a dialog box macro.

See also DB_CTRL_VERT_SCROLL@

DB_CTRL_INSET@

Displays an inset on a dialog box widget

Format DB_CTRL_INSET@(dbox, wdef, format ax_inset@ inset)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control
inset	An array of format ax_inset@

Description Displays a Graphics, Words or HTML inset on a canvas or Panel control. The ID argument must be the Control ID of one of these controls. The **ax_inset@ format** defines the inset and its display parameters.

Example

DB_CTRL_LABEL_COLOR@

Sets the color for a control's label

Format DB_CTRL_LABEL_COLOR@(dbox, ID, valueArray)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
valueArray	The color information as described below.

Description Sets the color for a control's label. The structure of the color information is as follows:

colorSpace	One of the following: WCSPACE#RGB 1 WCSPACE#HSB 5
colorValueArray	If the color space is RGB, a sub-array of three values corresponding to the red, green, and blue components of the color. If the color space is HSB, a sub-array of three values corresponding to the hue, saturation, and brightness components of the color.
name	A *nickname* for the color; for example, "magenta" or "red".

See also **DB_CTRL_GET_LABEL_COLOR@**
DB_CTRL_TEXT_COLOR@

DB_CTRL_WIDGET_COLOR@

DB_CTRL_LABEL_FONT@

Defines the font used to display a label

Format	DB_CTRL_LABEL_FONT@(dbox, ID, fontFamily)	
Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the control.
	font	A string naming one of the fonts that is known to Applix*ware*.

Description Names the font used to display a control's label.

See also DB_CTRL_GET_LABEL_FONT@
DB_CTRL_LABEL_FONT_SIZE@
DB_CTRL_LABEL_FONT_SLANT@
DB_CTRL_LABEL_FONT_WEIGHT@
DB_CTRL_LABEL_SHADOW@

DB_CTRL_LABEL_FONT_SIZE@

Defines a label's point size

Format	DB_CTRL_LABEL_FONT_SIZE@(dbox, ID, pointSize)	
Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the control.
	pointSize	The point size to be used.

Description Defines the point size of the text that will be used when a control's label is displayed. The default point size is 14 point.

See also DB_CTRL_GET_LABEL_FONT_SIZE@
DB_CTRL_LABEL_FONT@
DB_CTRL_LABEL_FONT_SLANT@
DB_CTRL_LABEL_FONT_WEIGHT@
DB_CTRL_LABEL_SHADOW@

DB_CTRL_LABEL_FONT_SLANT@

Defines the slant at which a label is displayed

Format DB_CTRL_LABEL_FONT_SLANT@(dbox, ID, slantVal)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
slantVal	The slant value, as follows:

0	Unslanted
1	Slanted (italics)

Other values are usually undefined. (A font could have additional slant values; however, none are provided with Applix*ware*.)

Description Specifies if a label's text is drawn at a slant. (More precisely, this macro indicates if an *emphasis* font such as *italic* is used instead of the normal font.)

See also DB_CTRL_GET_LABEL_FONT_SLANT@

DB_CTRL_LABEL_FONT@

DB_CTRL_LABEL_FONT_SIZE@

DB_CTRL_LABEL_FONT_WEIGHT@

DB_CTRL_LABEL_SHADOW@

DB_CTRL_LABEL_FONT_WEIGHT@

Defines the weight at which a label is displayed

Format DB_CTRL_LABEL_FONT_WEIGHT@(dbox, ID, weight)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
weight	The weight value, as follows:

0	Unweighted
1	Weighted (bold)

Other values are usually undefined. (A font could have additional weight values; however, none are provided with Applix*ware*.

Description Specifies if a label's text is drawn with additional weight. (More precisely, this macro indicates if an *emphasis* font such as **bold** is used instead of the normal font.)

See also DB_CTRL_GET_LABEL_FONT_WEIGHT@

DB_CTRL_LABEL_FONT@

DB_CTRL_LABEL_FONT_SIZE@

DB_CTRL_LABEL_FONT_SLANT@

DB_CTRL_LABEL_SHADOW@

DB_CTRL_LABEL_SHADOW@

Specifies if label's text is drawn with a shadow

Format DB_CTRL_LABEL_SHADOW@(dbox, ID, shadowFlag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
shadowFlag	A Boolean value where TRUE means to draw the characters using a drop shadow.

Description Specifies if label text is drawn using drop shadow letters.

DB_CTRL_LABEL_FONT@

DB_CTRL_LABEL_FONT_SIZE@

DB_CTRL_LABEL_FONT_SLANT@

DB_CTRL_LABEL_FONT_WEIGHT@

DB_CTRL_LENGTH@

Sets the number of characters typed into an entry box

Format DB_CTRL_LENGTH@(dbox, ID, numChars)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
numChars	The number of characters that can be typed in the entry box. If value is greater than the width of the entry box, the entry will scroll horizontally when the end of the entry box is reached.

Description Sets the number of characters that a user can type into an entry box. Normally, this value is set using the Dialog Box Editor. However, you can use DB_CTRL_LENGTH@ to set or change the number of characters that can be typed.

See also DB_CREATE_CTRL@

DB_CTRL_LINE_THICKNESS@

Specifies the line thickness for panel/line

Format DB_CTRL_LINE_THICKNESS@(dbox, ID, thickness)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the panel or line.
thickness	Indicates the thickness of the panel/line decoration. thickness can be in the range of -9 to 10. A negative thickness makes the panel/line appear recessed when the control display style is set to 3-dimensional. A thickness of -1 causes a thin recessed line. A thickness of -9 causes a thick recessed line. A positive thickness makes the panel/line appear raised when the control display style is set to 3-dimensional. A thickness of 1 causes a thin raised line. A value of 10 causes a thick raised line. A thickness of 0 specifies a thin line that is neither raised or recessed. The default is 0.

Description Sets a line's thickness and direction. Normally, the line thickness specification for panel/line decorations is made using the Dialog Box Editor. You can use DB_CTRL_LINE_THICKNESS@ to set or change the specification.

If the control display style is set to 2-dimensional, thickness affects line thickness but the panel/line does not appear recessed or raised.

See also DB_CREATE_CTRL@

DB_CTRL_MAX_VALUE@

Sets a scale's maximum value

Format DB_CTRL_MAX_VALUE@(dbox, ID, maxValue)

Arguments

dbox	The name of the dialog box variable.

	ID	The string name of the panel or line.
	maxValue	The scale's maximum value.

Description Sets the maximum value for a scale.

See also **DB_CTRL_MIN_VALUE@**

DB_CTRL_MIN_VALUE@

Sets a scale's minimum value

Format DB_CTRL_MIN_VALUE@(dbox, ID, minValue)

Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the panel or line.
	minValue	The scale's minimum value.

Description Sets the minimum value for a scale.

See also **DB_CTRL_MAX_VALUE@**

DB_CTRL_MONOSPACE@

Indicates whether list box items display in monospace typeface

Format DB_CTRL_MONOSPACE@(dbox, ID, flag)

Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the control.
	flag	Indicates whether list box items should be displayed using a monospace (Courier) typeface. TRUE indicates a monospace typeface for the list box items. FALSE indicates that list box items display using a proportional typeface.

Description Specifies that a monospace font will be used in a list box. Using a monospace typeface is helpful when including columns in your list box text. With the monospace typeface, all characters have equal width so you can align columns using the space bar.

DB_CTRL_MULTI_SELECT@

Specifies whether multiple selections are allowed in a list box

Format DB_CTRL_MULTI_SELECT@(dbox, ID, flag)

Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the control.
	flag	Indicates whether to allow multiple selections in the list box. If flag is TRUE, multiple selections are allowed. If flag is FALSE, only single selections are allowed in the list box. The default is FALSE.

Description Specifies whether multiple selection can occur in a list box. Normally, the specification to allow multiple selections in a list box is made using the Dialog Box Editor. You can use DB_CTRL_MULTI_SELECT@ to set or change this specification.

If multiple selections are not allowed, the first item in a list box is selected by default. If multiple selections are allowed, no items are selected by default. Default selections are initialized using DB_CTRL_VALUE@.

The way in which you select more than one item in a list box is to hold down the CTRL key while clicking the mouse to select more than one item from the list box.

The value of a list box that allows multiple selections is NULL if no item is selected, or it is an array that contains the indices of the marked items. The size of the array cannot exceed the number of marked items.

Note This macro also enables *click-ahead* on a push button. If you set this attribute on a button, it allows repeated clicks on a button rather than requiring waits between clicks.

DB_CTRL_MUX_TOGGLE@

Rotates a toggle button 45°

Format DB_CTRL_MUX_TOGGLE@(dbox, ID)

Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the control.

Description Rotates a toggle button 45° so that it stands on edge, like a hollow ``diamond." Without this macro, a toggle button lays flat, with two sides parallel to the baseline. There is no on/off switch for DB_CTRL_MUX_TOGGLE@; just remove the call if you want to turn the switch off.

DB_CTRL_NO_ECHO@

Suppress display of what the user types in an entry box

Format DB_CTRL_NO_ECHO@(dbox, ID, flag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
flag	A Boolean value which if set to TRUE indicates that the text typed in by the user is not redisplayed. Instead an asterisk (*) displays.

Description Suppresses the display of what the user types in an entry box. Normally, this macro is used in password type functions so that the information typed is not displayed. This entry box state cannot be set within the dialog box editor.

See also DB_CTRL_LENGTH@

DB_CTRL_OPTIONAL@

DB_CTRL_NO_TITLE@

Specifies whether an entry box title is displayed

Format DB_CTRL_NO_TITLE@(dbox, ID, flag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
flag	Indicates whether to display a title with the entry box. If flag is TRUE, no title is displayed with the entry box. If flag is FALSE, the entry box title is displayed as a label to the left of the entry box. The default is FALSE.

Description Sets or changes the display of a title. Not displaying a title is useful if you have a block of related entry boxes for which only a single title is necessary. Normally, this state is set using the Dialog Box Editor.

See also DB_CTRL_GET_TITLE@

DB_CTRL_TITLE@

DB_CTRL_OPTIONAL@

Specifies if an entry box entry is optional or mandatory

Format DB_CTRL_OPTIONAL@(dbox, ID, flag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
flag	Indicates whether an entry box entry is optional or mandatory. If flag is TRUE, the entry is optional. If flag is FALSE, the entry is mandatory. The default is FALSE.

Description Indicates if a text entry field is optional or mandatory. In most cases, you will set this value using the Dialog Box Editor. Because it is usually set in the Dialog Box Editor, the main function of this macro is to allow you to change the original specification.

If DB_CTRL_OPTIONAL@ is FALSE for an entry box (that is, entry is mandatory), an error message is displayed if someone attempts to exit from the dialog box without entering a value in this control.

See also DB_CTRL_GET_OPTIONAL@

DB_CTRL_LENGTH@

DB_CTRL_NO_ECHO@

DB_CTRL_PASTE@

Pastes the clipboard at the edit box cursor

Format DB_CTRL_PASTE@(dbox)

Arguments

dbox	The name of the dialog box variable.

Description Pastes the contents of the dialog-box clipboard at the current cursor position in an edit box. Text can be cut or copied from an edit box using DB_CTRL_CUT@ or DB_CTRL_COPY@

DB_CTRL_PICKABLE@

Indicates whether the items in a list box or edit box can be selected

Format DB_CTRL_PICKABLE@(dbox, ID, flag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
flag	Indicates whether items in a dialog box can be selected by the user. TRUE indicates that items can be selected. FALSE indicates that items cannot be selected.

Description Sets (or unsets) a dialog box control state that allows list box and edit box items to be selected. DB_CTRL_PICKABLE@ is useful when you want to use either these controls to display a scrollable list of informational text rather than a list of selectable items.

Using list boxes is simpler than using edit boxes. However, edit boxes are better at managing your application's memory.

If this macro is used with an edit box control, it sets the control's "read-only" flag.

See also DB_CTRL_PICK_DEFAULT@

DB_CTRL_PICK_DEFAULT@

Lets a double-click execute the default push button

Format DB_CTRL_PICK_DEFAULT@(dbox, ID, flag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
flag	Indicates whether a double-click of the mouse button selects the default push button. If flag is TRUE, double-clicking on a list box item selects the item and selects the default push button. The default is FALSE (which selects the item, but does not select the push button).

Description Specifies whether double-clicking on a list box item will select both the item and the default push button. Normally, this dialog box attribute is set using the Dialog Box Editor. If it is set there, you can use DB_CTRL_PICK_DEFAULT@ to set or change the specification.

If you use DB_CTRL_DISPLAY@ to change a push button's type to "hidden", it "pick-default" attribute is automatically set to FALSE. If you want the button to remain the default choice, move the button so it is outside the dialog box's frame. In this way, the button will remain clickable; however, it will not be drawn.

See also DB_CTRL_BUTTON_TYPE@

DB_CTRL_PICKABLE@

DB_CTRL_PIXMAPS@

Sets pixmaps for row and column pull downs

Format DB_CTRL_PIXMAPS@(dbox, ID, pixmapArray)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control
pixmapArray	The filenames of the pixmaps that will be displayed.

Description Sets pixmaps for row and column pulldowns.

See also DB_CTRL_ALTERNATE_PIXMAP@

DB_CTRL_RENDER_GRAPHIC@

Renders a graphic on a widget of the dialog box

Format DB_CTRL_RENDER_GRAPHIC@(dbox, ID, gfx, format gr_inset@ inset)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
gfx	The graphics handle.
inset	Information defining the graphic.

Description Renders a graphic on a widget of the dialog box. This macro forces the display of the graphic.

The definition of the gr_inset@ format is as follows:

```
format gr_inset@
        clear_mode,         '"none", "widget", "window"
        scale_mode,         `"clip to fit", "scale to fit", scale if clips"
        proportional,       'Boolean
```

```
prescale_x,           '100 is 1X
prescale_y,           '100 is 1X
format gr_area@ src,        'source position and size in current graphic units
format gr_area@ clip        'destination position and size in current graphic units
display_mode,         'special display effects
toggle_wait_cursor, 'should the inset display a wait cursor
page_numer,           'which page in the graphic to inset
widget                'widget to draw onto
```

See also DB_CTRL_ASSIGN_GRAPHIC@

DB_CTRL_RETURN_ON_CHANGE@

Suspends the operation of a dialog box so that other activities can be performed

Format DB_CTRL_RETURN_ON_CHANGE@(dbox, ID, flag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
flag	Indicates whether a control suspends dialog box operation when it is changed. A flag value of TRUE indicates that dialog box operation is suspended when the control is changed. FALSE indicates that dialog box operation is not suspended when the control is changed. The default is FALSE.

Description When the value of one control affects the value or display of another, you need to suspend the dialog box operation using DB_CTRL_RETURN_ON_CHANGE@ to make the appropriate value or display changes to the other controls. When you suspend operation of a dialog box, a dialog box exit condition is triggered. The exit condition is attributed to the control that suspends dialog box operation.

For entry boxes, use DB_CTRL_TYPING_RETURN@ or DB_CTRL_ACTIVE_-RETURN@ rather than DB_CTRL_RETURN_ON_CHANGE@ to suspend operation of a dialog box.

See also DB_CTRL_ACTIVE_RETURN@
DB_CTRL_TYPING_RETURN@

DB_CTRL_REVERSE_BITMAP@

Specifies the bitmap to use when a bitmap button is *selected*

Format DB_CTRL_REVERSE_BITMAP@(dbox, ID, pixmap)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
pixmap	The pixmap filename.

Description Specifies which bitmaps to use when bitmap buttons are pressed. When a pixmap button is pressed, the user expects that button to have a different look. On monochrome systems, the desired look can be achieved by inverting the bits in the pixmap. However, the visual appearance for color pixmaps must be indicated by replacing one pixmap with another using this macro.

DB_CTRL_RIGHT_MOUSE_MENU@

Defines menu displayed when right mouse button is clicked

Format DB_CTRL_RIGHT_MOUSE_MENU@(dbox, ID, ARRAYOF format rminfo@ info)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
info	An array of rminfo@ formats. The rminfo@ format is defined in dialog_.am. Each rminfo@ format contains the following information:

```
format rminfo
    name,          'The text to be displayed in the menu
    macro_name,
                   'The name of the macro called when the menu item is se-
                   lected
    active         `A Boolean value indicating if the item is enabled (normal
                   text) or disabled (grayed text).
```

Description Displays one or more menu items above a text control when then right mouse button is pressed. Your mouse pointer must be over the control before you press the right mouse button in order for the menu to appear. This feature works only with entry boxes, list boxes, tables, and edit boxes.

Example

DB_CTRL_SHADOW@

Indicates if drop shadow is drawn behind a button

Format DB_CTRL_SHADOW@(dbox, ID, flag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
flag	A Boolean value that when set to TRUE indicates that a drop shadow is drawn behind a button.

DB_CTRL_STRINGS@

Sets string values for list boxes, radio buttons, and option button controls

Format DB_CTRL_STRINGS@(dbox, ID, value)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
value	A string or string array.

Description Specifies the string values to be displayed in controls. In most cases, you will use this macro for specifying values for list boxes. If you specify a string array for value, each string in the array is displayed on a separate line in the list box.

You can also use DB_CTRL_STRINGS@ to indicate the text displayed next to radio buttons or to set the options displayed in an option button menu. However, radio button and option button values are normally set using the Dialog Box Editor.

To set a string in an edit box use **DB_EDITBOX_SET_DATA@**.

To set a string in an entry field use **DB_CTRL_VALUE@**.

To set a string in a dialog box title use **DB_CTRL_TITLE@**.

See also **DB_CREATE_CTRL@**

DB_CTRL_STYLE@

Sets the style of the row/column pull down control

Format DB_CTRL_STYLE@(dbox, ID, value)

Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the control.
	value	Sets one of the following display styles:

0	RC_STYLE#SQUARE	Cells arranged in a square (default)
1	RC_STYLE#COLUMN	Cells arranged in a single column
2	RC_STYLE#ROW	Cells arranged in a single row

Description Sets the style of the row/column widget.

DB_CTRL_TEXT_AND_FONT@

Places a string (in a named font) into a button or label

Format DB_CTRL_TEXT_AND_FONT@(dbox, ID, text, font)

Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the control.
	text	The text to display.
	font	The font in which the text is displayed.

Description Places a string using the named font into a button or label.

Example

DB_CTRL_TEXT_COLOR@

Sets the text's color

Format DB_CTRL_TEXT_COLOR@(dbox, ID, valArray)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
valueArray	The color information as described below.

Description Sets the color for a control's text; that is, this the text the user types. In an entry field, this is the text the user types. The structure of the color information is as follows:

colorSpace	One of the following: WCSPACE#RGB 1 WCSPACE#HSB 5
colorValueArray	If the color space is RGB, a sub-array of three values corresponding to the red, green, and blue components of the color. If the color space is HSB, a sub-array of three values corresponding to the hue, saturation, and brightness components of the color.
name	A *nickname* for the color; for example, "magenta" or "red".

See also DB_CTRL_GET_TEXT_COLOR@

DB_CTRL_LABEL_COLOR@

DB_CTRL_WIDGET_COLOR@

DB_CTRL_TEXT_FONT@

Defines the font used to draw text

Format DB_CTRL_TEXT_FONT@(dbox, ID, fontName)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
fontName	The name of a font currently defined within the Applix*ware* environment.

Description Names the font that will be used to display characters when the user enters text within an entry field.

See also	DB_CTRL_GET_TEXT_FONT@
	DB_CTRL_TEXT_FONT_SIZE@
	DB_CTRL_TEXT_FONT_SLANT@
	DB_CTRL_TEXT_FONT_WEIGHT@
	DB_CTRL_TEXT_SHADOW@

DB_CTRL_TEXT_FONT_SIZE@

Defines the point size used to draw text

Format DB_CTRL_TEXT_FONT_SIZE@(dbox, ID, pointSize)

Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the control.
	pointSize	The numeric point size to be used.

Description Defines the point size for the text that will be used to display characters when the user enters text within an entry field.

The box into which the user types information will be resized to conform to the new point size. Because the position of other controls is not changed, you may need to change how this and other controls are placed.

See also	DB_CTRL_GET_TEXT_FONT_SIZE@
	DB_CTRL_TEXT_FONT@
	DB_CTRL_TEXT_FONT_SLANT@
	DB_CTRL_TEXT_FONT_WEIGHT@
	DB_CTRL_TEXT_SHADOW@

DB_CTRL_TEXT_FONT_SLANT@

Defines the slant at which text is drawn

Format DB_CTRL_TEXT_FONT_SLANT@(dbox, ID, slant)

Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the control.
	slant	A number indicating the slant.

Description Defines the slant property used to when a font is drawn. More precisely, a slant number indicates that the italic font should be chosen. That is, this macro indicates that another font within the current font family is to be used. Normal values are 0 (no slant) and 1 (slanted or italic). While it is possible for other slant numbers to be used, no font provided by Applixware uses these values.

If the font indicated by a slant number is unavailable, the output results are undefined.

See also DB_CTRL_GET_TEXT_FONT_SLANT@

DB_CTRL_TEXT_FONT@

DB_CTRL_TEXT_FONT_SIZE@

DB_CTRL_TEXT_FONT_WEIGHT@

DB_CTRL_TEXT_SHADOW@

DB_CTRL_TEXT_FONT_WEIGHT@

Defines the weight (boldness) at which text is drawn

Format DB_CTRL_TEXT_FONT_WEIGHT@(dbox, ID, weight)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
weight	A number indicating the font weight.

Description Defines the weight property used to when a font is drawn. (The weightproperty controls "boldness".) More precisely, a weight number indicates that the bold font should be chosen. That is, this macro indicates that another font within the current font family is to be used. Normal values are 0 (no weight) and 1 (bold). While it is possible for other weights numbers to be used, no font provided by Applixware uses these values.

If the font indicated by a weight number is unavailable, the output results are undefined.

See also DB_CTRL_GET_TEXT_FONT_WEIGHT@

DB_CTRL_TEXT_FONT@

DB_CTRL_TEXT_FONT_SIZE@

DB_CTRL_TEXT_FONT_SLANT@

DB_CTRL_TEXT_SHADOW@

DB_CTRL_TEXT_SHADOW@

Indicates if entry field text is drawn with drop shadow

Format DB_CTRL_TEXT_SHADOW@(dbox, ID, flag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
flag	A Boolean value which if set to TRUE indicates that drop shadow letters are used.

Description Specifies that the letters typed by a user in an entry field are drawn with a drop shadow.

See also DB_CTRL_GET_TEXT_SHADOW@

DB_CTRL_TEXT_FONT@

DB_CTRL_TEXT_FONT_SIZE@

DB_CTRL_TEXT_FONT_SLANT@

DB_CTRL_TEXT_FONT_WEIGHT@

DB_CTRL_THREE_STATE_TOGGLE@

Allows a toggle to be a three-state toggle

Format DB_CTRL_THREE_STATE_TOGGLE@(dbox, ID, flag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
flag	A Boolean value where TRUE indicates that the toggle control will be a three-state toggle.

Description Indicates that a toggle will be a three-state toggle (on, off, and gray) instead of a two-state toggle (on and off).

The values returned by DB_CTRL_GET_VALUE@ for a three-state toggle are as follows:

0	Off
1	On
2	Grayed

DB_CTRL_TITLE@

Sets the title for a control

Format DB_CTRL_TITLE@ (dbox, ID, string)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
string	A string indicating the title for the control.

Description Specifies the title for a dialog box control. Normally, the title for a control is specified using the Dialog Box Editor. You can use DB_CTRL_TITLE@ to set or change the title for a control.

To set a string in a list box use **DB_CTRL_STRINGS@**.

To set a string in an edit box use **DB_EDITBOX_SET_DATA@**.

To set a string in an entry field use **DB_CTRL_VALUE@**.

Example

See also **DB_CTRL_GET_TITLE@**
DB_CTRL_NO_TITLE@
DB_CTRL_TITLE_TYPE@

DB_CTRL_TITLE_TYPE@

Sets a control's title positioning

Format DB_CTRL_TITLE_TYPE@(dbox, ID, value)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
value	One of the following values:

-1	No title
0	Left title
1	Top title

Description Determines the control's title positioning.

See also DB_CTRL_GET_TITLE@
DB_CTRL_NO_TITLE@
DB_CTRL_TITLE@

DB_CTRL_TRIM@

Specifies whether leading and trailing blank spaces are removed

Format DB_CTRL_TRIM@ (dbox, ID, flag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
flag	A Boolean value where TRUE means that spaces are trimmed.

Description Specifies if leading and trailing spaces in the values entered within entry boxes are removed. Normally, this value is set within the Dialog Box Editor.

By default, spaces are trimmed.

DB_CTRL_TYPING_RETURN@

Sends control back to the invoking macro when a character is typed

Format DB_CTRL_TYPING_RETURN@ (dbox, ID, flag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
flag	Indicates whether dialog box operation should be suspended when a character is typed. TRUE indicates that dialog box operation is suspended. FALSE indicates that dialog box operation is not suspended. The default is FALSE.

Description Sends execution control back to the invoking macro when a character is typed in an entry box. In this way, your macro can directly respond to the characters being typed by the user.

See also DB_CTRL_ACTIVE_RETURN@
DB_CTRL_RETURN_ON_CHANGE@

DB_CTRL_VALID_CHARS@

Specifies the characters that can be typed

Format DB_CTRL_VALID_CHARS@ (dbox, ID, validChars)

Arguments		
	dbox	The name of the dialog box variable.
	ID	The string name of the control.
	validChars	A string containing the characters that are allowable entries in the entry box.

Description Specifies the characters a user can type while entering information into an entry box. If a character other than those specified by DB_CTRL_VALID_CHARS@ is typed, the system bell sounds and the character is not accepted.

DB_CTRL_VALID_CHARS@ should only be used in situations where it is obvious that certain characters are not valid entries for an entry box. DB_CTRL_VALID_CHARS@ does not provide any indication of what is and isn't a valid character other than to beep when an invalid character is entered.

DB_CTRL_VALUE@

Sets the value for a dialog box control

Format DB_CTRL_VALUE@ (dbox, ID, value)

Arguments		
	dbox	The name of the dialog box variable.
	ID	The string name of the control.
	value	The value to set the control, which is one of the following types:

toggle buttons

A toggle button can be TRUE or FALSE. TRUE indicates that the toggle button is chosen. FALSE indicates that the toggle button is not chosen. If you do not initialize a toggle button value, value is FALSE (not selected).

A three-state toggle can be set to one of the following values:

0	Off
1	On
2	Grayed

entry boxes	A string indicating the text to be displayed in the entry field when the dialog box is displayed. If you did not initialize an entry box value, nothing is displayed in the entry box.

radio button groups, option buttons, and list boxes

A number indicating the option that is selected. For radio button groups and option buttons, the numbering of options corresponds to the order you list the options when you define the control using the Dialog Box Editor.

For list boxes, the numbering of options is based on the option's position in the array in which the options were defined, unless multiple selections are allowed. If multiple selections are allowed, value must be an array in which each element corresponds to a list box selection.

If you do not initialize a value, item 0 is selected when the dialog box is displayed.

Description Initializes a control value. Any control initialization statements must appear after DB_LOAD@ and before DB_DISPLAY@ in a dialog box macro. If you do not initialize control values, then the control values are displayed as follows:

- Entry fields appear blank
- Toggle buttons are not selected
- The first item in a list box is selected. If multiple selections are allowed, nothing is selected.
- For radio buttons and option buttons, the item that was defined as item 0 when the control was created is selected.

To set a string in a list box use DB_CTRL_STRINGS@.

To set a string in an edit box use DB_EDITBOX_SET_DATA@.

To set a string in a dialog box title use DB_CTRL_TITLE@.

Example

DB_CTRL_VERT_SCROLL@

Controls the vertical scrollbar in list boxes

Format DB_CTRL_VERT_SCROLL@ (dbox, ID, flag)

Arguments

dbox	The name of the dialog box variable.

ID	The string name of the control.
flag	A Boolean value where TRUE means that the vertical scrollbar will be displayed.

Description Controls the display of vertical scroll bar in list boxes**.**

See also **DB_CTRL_HORIZ_SCROLL@**

DB_CTRL_VISIBLE_ITEM_COUNT@

Sets the number of visible table lines

Format DB_CTRL_VISIBLE_ITEM_COUNT@(dbox, ID, num)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
num	The number of lines that will be displayed.

Description Sets the number of table rows that are displayed within the table widget. This number does not include the heading line.

DB_CTRL_WIDGET_COLOR@

Sets the control's color

Format DB_CTRL_WIDGET_COLOR@(dbox, ID, valueArray)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
valueArray	The color information as described below.

Description Sets the color for a control. The structure of the color information is as follows:

colorSpace	One of the following: WCSPACE#RGB 1 WCSPACE#HSB 5
colorValueArray	If the color space is RGB, a sub-array of three values corresponding to the red, green, and blue components of the color. If the color space is HSB, a sub-array of three values corresponding to the hue, saturation, and brightness components of the color.

name	A *nickname* for the color; for example, "magenta" or "red".

See also DB_CTRL_TEXT_COLOR@
DB_CTRL_LABEL_COLOR@
DB_CTRL_GET_WIDGET_COLOR@

DB_CTRL_WIDTH@

Sets the display width for a dialog box control

Format DB_CTRL_WIDTH@ (dbox, ID, width)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
width	For entry boxes, push buttons, and list boxes, width is a number indicating the width, in characters, to which to set the control. For edit boxes and panel/line decorations, width indicates the width in pixels.

Description Sets the display width for a dialog box control. Normally, the display width for entry boxes, push buttons, list boxes, and panel/lines is set using the Dialog Box Editor. You can use DB_CTRL_WIDTH@ to set or change the width.

See also DB_CREATE_CTRL@
DB_CTRL_GET_WIDTH@
DB_CTRL_HEIGHT@

DB_CTRL_WORK_COLORS@

Specifies whether a control is displayed in work area colors instead of control panel colors

Format DB_CTRL_WORK_COLORS@ (dbox, ID, flag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
flag	Indicates whether work area colors should be used to display the control. If flag is TRUE, the control is displayed using work area colors. If flag is

FALSE, the control is displayed using control panel colors. The default is FALSE.

Description Used to contrast some controls from the dialog box background. Only list boxes and edit boxes may be displayed in work area colors. DB_CTRL_WORK_COLORS@ has no effect if the Force all text Entry Widgets to use Work Area Background Color preference is TRUE.

DB_CTRL_XPOS@

Sets the X-position of a dialog box control

Format DB_CTRL_XPOS@(dbox, ID, xPos)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
xPos	A number indicating the x-axis location, in pixels, of the top left corner of the control. xPos indicates the pixel position of the top left corner of the control with reference to the top left corner of the dialog box (position (0,0)).

Description Normally, the location of a control is specified using the Dialog Box Editor. You can use DB_CTRL_XPOS@ to set or change the x-axis location for a control.

See also **DB_CTRL_GET_XPOS@**

DB_CTRL_YPOS@

DB_CTRL_YPOS@

Sets the Y-position of a dialog box control

Format DB_CTRL_YPOS@ (dbox, ID, yPos)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
yPos	A number indicating the y-axis location, in pixels, of the top left corner of the control. yPos indicates the pixel position of the top left corner of the control with reference to the top left corner of the dialog box (position (0,0)).

Description	Normally, the location of a control is specified using the Dialog Box Editor. You can use DB_CTRL_YPOS@ to set or change the y-axis location for a control.
See also	DB_CTRL_GET_YPOS@ DB_CTRL_XPOS@

DB_CURSOR_IN_ENTRY@

Places the cursor in an entry field

Format DB_CURSOR_IN_ENTRY@(dbox, ID, startChar, endChar)

Arguments

dbox	The name of a dialog box variable.
ID	The string name of the entry field.
startChar	The character at which highlighting will begin. The first character in the entry field is character 1.
endChar	The character at which highlighting will end.

Description Places the cursor in the entry field identified by ID. All characters from startChar to endChar are highlighted.

If startChar is set to TRUE, the entire field is highlighted.

If startChar is set to 0, ELF assumes that character numbering is zero-based instead of 1 based.

If endChar is less than 0 or is less than startChar, the entire control is highlighted.

See also DB_GET_CURSOR_IN_ENTRY@

DB_DESTROY_CTRL@

Removes a control

Format DB_DESTROY_CTRL@(dbox, ID)

Arguments

dbox	The name of a dialog box variable.
ID	The name of the control to be destroyed.

Description Removes a control from the dialog box and the display.

See also **DB_CREATE_CTRL@**

DB_DIALOG_CALLBACKS@

Associates a callback function with a dialog box

Format DB_DIALOG_CALLBACKS@(dbox, cbinfo)

Arguments

dbox	The name of a dialog box variable.
cbinfo	An array that takes the following form:

<<eventName1><funcName1>>[,
<<eventName2><funcName2>>[,
<<eventName3><funcName3>>[,
<<eventName4><funcName4>>]]]

The following values may be used for events:

8	Poke event
9	Resize event
10	Menu bar event
11	Dialog box initialization event

For example:

```
cbinfo[0,0] = 10
cbinfo[0,1] = "dd_menu_bar_event_func"
cbinfo[1,0] = 8
cbinfo[1,1] = "dd_poke_event_func"
```

Description Associates a callback function with a dialog box. That is, when one of the four events listed above occurs, the function associated with the event is invoked.

See also **DB_CTRL_CALLBACKS@**

DB_DISABLE_RECORD@

Disables ELF recording

Format DB_DISABLE_RECORD@(dbox, flag)

Arguments

dbox	The name of a dialog box variable.

	flag	If flag it set to TRUE, ELF recording is not allowed. If flag it set to FALSE, ELF recording is allowed

Description Turns on and off ELF recording.

DB_DISPLAY@

Displays a dialog box

Format DB_DISPLAY@(dbox)

Arguments dbox The name of a dialog box variable.

Description Paints a dialog box and makes it ready for processing. A dialog box file must be loaded using **DB_LOAD@** before it can be displayed using this macro.

Example

See also **DB_CLOSE@**
DB_DISPLAY_ONLY@

DB_DISPLAY_ONLY@

Displays a dialog box without making it ready for processing

Format DB_DISPLAY_ONLY@(dbox, flag)

Arguments	dbox	The name of a dialog box variable.
	flag	A Boolean value which if set to TRUE displays the dialog box without making it ready for processing. FALSE allows processing to occur.

Description Displays a dialog box without allowing any processing to occur. This is useful in the situations where you must acquire state information from the dialog box before allowing the user to use the box.

This macro is most often used in conjunction with edit box and table controls. These controls, unlike other controls, assume that control is displayed before data is written to it.

After you have acquired the information you need (by calling DB_DISPLAY_ONLY@ with a flag value of TRUE), you must allow processing to resume by reinvoking the DB_DISPLAY_ONLY@ macro with flag set to FALSE.

See also DB_CLOSE@
DB_DISPLAY@

DB_EDITBOX_CLEAR@

Deletes the contents of an edit box

Format DB_EDITBOX_CLEAR@(dbox, ID)

Arguments

dbox	The name of a dialog box variable.
ID	The string name of the control.

Description Removes the contents of an edit box.

See also DB_EDITBOX_GET_DATA@
DB_EDITBOX_GET_SELECTION@
DB_EDITBOX_SELECTION@
DB_EDITBOX_SET_DATA@

DB_EDITBOX_GET_DATA@

Retrieves the contents of an edit box

Format stringArray = DB_EDITBOX_GET_DATA@(dbox, ID)

Arguments

dbox	The name of a dialog box variable.
ID	The string name of the control.

Description Retrieves the contents of an edit box. This information is returned as an array with each element in the array corresponding to one element of the array used to initialize the editbox's contents.

See also DB_EDITBOX_CLEAR@
DB_EDITBOX_GET_SELECTION@
DB_EDITBOX_SELECTION@
DB_EDITBOX_SET_DATA@

DB_EDITBOX_GET_SELECTION@

Returns the selected region's row and column addresses

Format posArray = DB_EDITBOX_GET_SELECTION@(dbox, ID)

Arguments		
	dbox	The name of a dialog box variable.
	ID	The string name of the control.

Description Returns a four element array containing the row and column address of the selected region within an editbox, as follows:

- Starting row
- Starting column
- Ending row
- Ending column

See also DB_EDITBOX_CLEAR@
DB_EDITBOX_GET_DATA@
DB_EDITBOX_SELECTION@
DB_EDITBOX_SET_DATA@

DB_EDITBOX_SELECTION@

Marks a region in an editbox

Format DB_EDITBOX_SELECTION@(dbox, ID, startRow, startCol, endRow, endCol)

Arguments		
	dbox	The name of a dialog box variable.
	ID	The string name of the control.
	startRow	The starting row of the region.
	startCol	The starting column of the region.
	endRow	The ending row of the region.
	endCol	The ending column of the region

Description Marks a region in an editbox.

See also DB_EDITBOX_CLEAR@

DB_EDITBOX_GET_DATA@
DB_EDITBOX_GET_SELECTION@
DB_EDITBOX_SET_DATA@

DB_EDITBOX_SET_DATA@

Replaces the contents of the edit box

Format DB_EDITBOX_SET_DATA@(dbox, ID, stringArray)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
stringArray	An array of strings that replaces the existing contents of the editbox.

Description Replaces the editbox's contents. The new data must be supplied as an array of strings.

To set a string in a list box use **DB_CTRL_STRINGS@**.

To set a string in an entry field use **DB_CTRL_VALUE@**.

To set a string in a dialog box title use **DB_CTRL_TITLE@**.

See also **DB_EDITBOX_CLEAR@**
DB_EDITBOX_GET_DATA@
DB_EDITBOX_GET_SELECTION@
DB_EDITBOX_SELECTION@

DB_EXIT_CODE@

Returns the event type that caused an exit

Format num = DB_EXIT_CODE@(dbox)

Arguments

dbox	The name of the dialog box variable.

Description Returns the event type that caused an exit (with an exit control). This macro is only used for table controls.

The returned event type is one of the following:

EV_TBL_SELECTION_CHANGE_	50
EV_TBL_DBLCLICK_	51
EV_TBL_KEYPRESS_	55

EV_TBL_COLUMN_RESIZE_	56
EV_TBL_COPY_TO_CLIPBOARD_	58
EV_TBL_PASTE_FROM_CLIPBOARD_	59
EV_TBL_BUTTON_PRESS_	60

If you double-click a table row, the table control returns two exit codes to your ELF macro: EV_TBL_SELECTION_CHANGE_ (50) and EV_TBL_DBLCLICK_ (51). The click event (50) is always returned first. The double-click event (51) is returned second.

See also DB_EXIT_CTRL@

DB_EXIT_DATA@

DB_EXIT_INFO@

DB_EXIT_CTRL@

Returns the control name that triggered termination

Format ID = DB_EXIT_CTRL@(dbox)

Arguments dbox The name of the dialog box variable.

Description Returns a string indicating the control ID that enabled processing of your ELF macro that invoked the dialog box to continue. An "exit" condition occurs when a dialog box is exited or when a dialog box operation is suspended. This macro is useful for re-entrant dialog boxes in which you can specify a course of action based on exit condition.

For example, selecting an entry in the top Help list box invokes an exit condition. The processing continues to set values in the lower list box.

Example

See also DB_EXIT_CODE@

DB_EXIT_DATA@

DB_EXIT_INFO@

DB_EXIT_DATA@

Returns data associated with an exit

Format data = DB_EXIT_DATA@(dbox)

Arguments dbox The name of the dialog box variable.

Description Returns the data associated with an exit. The data returned is dependent on the kind of event type that initiated the action. (See DB_EXIT_CODE@ for more information.) The following list summarizes the exit data that can be returned for each event code.

50	EV_TBL_SELECTION_CHANGE_ An array containing the new selection
51	EV_TBL_DBLCLICK_ An integer containing the row number
56	EV_TBL_COLUMN_RESIZE_ A two-element array containing the column number of the column being resized and its new width in pixels
58	EV_TBL_COPY_TO_CLIPBOARD_
59	EV_TBL_PASTE_FROM_CLIPBOARD_
60	EV_TBL_BUTTON_PRESS_ A four element array containing the row number column number, starting character position, and ending character position

This macro is only used with table controls.

See also DB_EXIT_CODE@

DB_EXIT_CTRL@

DB_EXIT_INFO@

DB_EXIT_INFO@

Returns additional exit control information

Format info = DB_EXIT_INFO@(dbox)

Arguments dbox The name of the dialog box variable.

Description Returns additional information associated with an exit control. It is set at dialog box exit/suspend time. The values returned are as follows:

EV_CHANGED_
Control has new value
EV_FOCUS_IN_
Entry field/editbox received focus
EV_FOCUS_OUT_
Entry field/editbox lost focus
EV_TYPING_
Entry field per character callback
EV_DBLCLICK_
List boxes

EV_MULTI_SELECT_	List boxes only
EV_MOTION_	Scales
EV_STROKE_SELECT_	List boxes
EV_ARMED_	Mouse down or drag enter on push button
EV_DISARMED_	Mouse up or drag exit on push button
EV_DLG_RESIZED_	Dialog box was resized
EV_LIST_DROP_	

Tables do not use this macro. Instead, they use **DB_EXIT_DATA@**.

See also **DB_EXIT_CODE@**

DB_EXIT_CTRL@

DB_EXIT_DATA@

DB_EXPRESSLINE_STATUS@

Returns current display status of *Express*line

Format	flag = DB_EXPRESSLINE_STATUS@(dbox)
Arguments	dbox — The name of the dialog box variable.
Description	Returns the current display status of the *Express*line.

DB_GET_CTRL_NAMES@

Returns a list of control names

Format	nameArray = DB_GET_CTRL_NAMES@(dbox)
Arguments	dbox — The name of the dialog box variable.
Description	Returns an array containing the names of all controls defined in a dialog box.

DB_GET_CURSOR_IN_ENTRY@

Returns the entry box ID in which the cursor resides

Format	ID = DB_GET_CURSOR_IN_ENTRY@(dbox)
Arguments	dbox — The name of the dialog box variable.
Description	Returns the name of the entry box in which the cursor resides when dialog box execution is terminated.
See also	DB_CURSOR_IN_ENTRY@

DB_GET_HEIGHT@

Returns the height of a dialog box

Format	height = DB_GET_HEIGHT@(dbox)
Arguments	dbox — The name of the dialog box variable.

Description Returns an array of two elements. The first element is the height of the dialog box on a workstation running X-Windows Motif. The second element is the height of the dialog box on a workstation running Open Windows.

See also DB_GET_WIDTH@

DB_HEIGHT@

DB_GET_ICON_TITLE@

Returns the title used when a window is displayed as an icon

Format title = DB_GET_ICON_TITLE@(dbox)

Arguments dbox The name of the dialog box variable.

Description Returns a string that is the text of the title which is displayed when a window is mimimized-that is, when the window is changed into a set-aside icon.

See also DB_ICON_TITLE@

DB_GET_TITLE@

DB_GET_POKE@

Returns the most recently received poke code

Format pokeNum = DB_GET_POKE@(dbox)

Arguments dbox The name of the dialog box variable.

Description Returns the code of the most recently received poke message. You can use this value to determine how a dialog box should respond to the message. When a message (sent using DB_SEND_POKE@) is received, the exit code poke_ is triggered.

You should only use DB_GET_POKE@ to return the poke code after you determine that the poke_ exit code was received.

See also DB_GET_POKE_DATA@

DB_GET_POKE_DATA@

Returns the poke message string

Format message = DB_GET_POKE_DATA@(dbox)

Arguments dbox The name of the dialog box variable.

Description Returns the message associated with a poke code. You should only use DB_GET_POKE_DATA@ to return a message string after you determine that the poke_ exit code was sent. If your dialog box can receive more than one poke_ code, use **DB_GET_POKE@** to determine which poke code was sent.

DB_GET_POKE_LIST@

Returns the poke codes being received by a dialog box

Format pokeArray = DB_GET_POKE_LIST@(dbox)

Arguments dbox The name of the dialog box variable.

Description Returns an array of integer constants that represent the poke values associated with the dialog box. These values were initially set using **DB_ACCEPT_POKES@**.

DB_GET_TITLE@

Returns the dialog box title

Format title = DB_GET_TITLE@(dbox)

Arguments dbox The name of the dialog box variable.

Description Returns a string containing the title of the dialog box identified by dbox.

See also **DB_TITLE@**

DB_GET_WIDTH@

Returns the width of a dialog box

Format	width = DB_GET_WIDTH@(dbox)	
Arguments	dbox	The name of the dialog box variable.

Description Returns an array of two elements. The first element is the width of the dialog box on a workstation running X-Windows Motif. The second element is the width of the dialog box on a workstation running Open Windows.

See also DB_GET_HEIGHT@
DB_WIDTH@

DB_HEIGHT@

Sets the height of a dialog box

Format	DB_HEIGHT@(dbox, height)	
Arguments	dbox	The name of the dialog box variable.
	height	The height, in pixels, to make the dialog box.

Description Normally, the dialog box height is set using the Dialog Box Editor. If you set the height using DB_HEIGHT@, make sure that the height is sufficient to contain all the controls in the dialog box.

See also DB_GET_HEIGHT@
DB_WIDTH@

DB_HELP_TOPIC@

Associates help text with a particular dialog box

Format	DB_HELP_TOPIC@(dbox, hypertargetName)	
Arguments	dbox	The name of the dialog box variable.

hypertargetName
The name of a hypertarget field within any of the files within your hypertext system.

Description Defines the help topic to be associated with dialog box. The default topic name is the dialog box name. Because the default topic name is the dialog box name, you only need use this command when you wish to change the default associations. For example, if you are using the same dialog box in two different contexts (as is, for example, the File ® Open dialog box within Applix*ware*), you can change the association so that help relevent to the current context appears.

A second way to use this command is when you want the help for more than one dialog box to point to the same help text

DB_ICON@

Specifies a dialog box's set-aside icon

Format DB_ICON@(dbox, bitmapNum)

Arguments

dbox	The name of the dialog box variable.
bitmapNum	A number corresponding to the numeric file name you gave the bitmap file when you saved it.

Description Specifies the icon to be used as the set-aside icon for a dialog box. You can create the graphic for a set-aside icon using the Bitmap Editor. The size of the created icon size should be 16x16, 32x32, 50x50, or 64x64 pixels.

The file name containing the bitmap must use the following naming scheme:

· The first part of the bitmap file must be a numeric name. For example, you could name the bitmap file "34."

· The second part of the bitmap file must contain one of the following strings that indicated the size of the bitmap:

-16x16
-32x32
-50x50
-64x64

The bitmapNum is the number used as the first part of the bitmap file name. When DB_ICON@ looks for the bitmap file, it automatically appends a string corresponding to the Icon size setting in the Customize Look and Feel dialog box to bitmap. If you want a set-aside icon to be used regardless of the Icon size setting, you must create four different icons corresponding to the four possible sizes.

DB_ICON_TITLE@

Defines a set-aside icon's title

Format DB_ICON_TITLE@(dbox, title)

Arguments

dbox	The name of the dialog box variable.
title	The icon's title.

Description Sets the text that is displayed at the bottom of a set-aside icon. (A *set-aside* icon is the image displayed when a window in minimized.)

See also **DB_GET_ICON_TITLE@**
DB_TITLE@

DB_ICONIZE@

Changes a window into an icon or vice-versa

Format flag = DB_ICONIZE@(dbox, iconFlag)

Arguments

dbox	The name of the dialog box variable.
iconFlag	A Boolean value which if set to TRUE changes the window into a set-aside icon. FALSE changes a set-aside icon into a window.

Description If iconFlag is set to TRUE, the current Applix*ware* window is changed into a set-aside icon. If the operation is successful, TRUE is returned. (It would be unsuccessful if the window were already being displayed as a set-aside icon.)

Similarly, setting iconFlag to FALSE changes a set-aside icon into a window.

See also **DB_IS_SETASIDE@**
DB_ICON_TITLE@

DB_IS_SETASIDE@

Returns Boolean indicating if a window is being displayed as an icon

Format flag = DB_IS_SETASIDE@(dbox)

Arguments	dbox	The name of the dialog box variable.

Description Returns a Boolean value indicating if the Applix*ware* window is being displayed as an icon or a window. TRUE means that the window is being displayed as an icon.

See also DB_ICONIZE@

DB_LOAD@

Loads dialog box information

Format DB_LOAD@(dboxFile[, forceReadFlag])

Arguments	dboxFile	A string indicating the file name of a dialog box layout file that was created with the Dialog Box Editor or with DB_CREATE_DIALOG@. The .d extension can be either included or excluded. The macro works both ways.
	forceReadFlag	A Boolean value which if set to TRUE tells ELF that if a dialog box is loaded into memory more than once, it should reread the dialog box's definition from disk rather than from the Applix*ware* memory cache.

Description Loads a dialog box definition into memory. The first time DB_LOAD@ is executed in a macro, the specified dialog box file is loaded from disk. For all subsequent executions of DB_LOAD@, the dialog box layout information is retrieved from memory.

Example

See also DB_CREATE_DIALOG@

DB_MENU_BAR@

Loads a menu bar in a dialog box

Format DB_MENU_BAR@(dbox, ID)

Arguments	dbox	The name of the dialog box variable.
	ID	The ID number of a menu bar defined by SET_SELECTIONS@.

Description Loads a menu bar that has been stored in memory using SET_SELECTIONS@. DB_MENU_BAR@ must be placed before DB_DISPLAY@ in your dialog box macro.

DB_MENU_BAR_WORD@

Returns the macro associated with a menu bar command

Format macroName = DB_MENU_BAR_WORD@(dbox)

Arguments

dbox	The name of the dialog box variable.

Description Returns a string indicating the macro called by the menu option that was selected from a dialog box menu bar. It is up to you to specify the action to perform based on the macro returned. In most cases, you will simply execute the macro.

DB_MENU_STATUS@

Sets toggle and graying attributes for menu bar options

Format DB_MENU_STATUS@(dbox, name, value)

Arguments

dbox	The name of the dialog box variable.
name	The name (as a string) of the macro that is called by the menu option for which you want to specify an attribute.
value	The attribute to set. The header file containing the DEFINE statements for these value values is menubar_.am:

MENUSTAT#NORMAL
: Menu option is displayed normally.

MENUSTAT#DIMMED
: Menu option is grayed.

MENUSTAT#TOGGLE_ON
: Menu option is toggled on. The toggle appears to the left of the menu option in the pull-down menu.

MENUSTAT#TOGGLE_OFF
: Menu option is made a toggle option, but the option is toggled off.

MENUSTAT#RADIO_OFF
: Menu option is a toggle option and a radio button is used as the toggle. The radio button is toggled off.

MENUSTAT#RADIO_ON
: Menu option is a toggle option and a radio button is used as the toggle. The radio button is toggled on, to the left of the menu option in the pull-down menu.

MENUSTAT#NO_SHOW
Menu option is not displayed on the pull-down menu.

Description Sets attributes for menu options in dialog boxes you have created. The header file menubar_.am contains constant definitions for the menu status values. If you want to use these definitions rather than numbers for value, include the statement INCLUDE "menubar_.am" in your macro.

DB_MENU_STATUS@ must appear after **SET_SELECTIONS@** and before **DB_DISPLAY@**.

DB_OWNERLESS@

Makes a dialog box macro into a top-level ELF task

Format DB_OWNERLESS@(dbox, flag)

Arguments		
	dbox	The name of a dialog box variable
	flag	If TRUE, it makes the current dialog box macro a top-level ELF task. This means that the macro is terminated only by exiting on its own, or exiting out of Applix*ware*. If a macro is not a top-level task, the macro is terminated when its macro parent task exits.

Description The DB_OWNERLESS@ macro is used by the main Applix*ware* menu. It makes a dialog box into a top-level ELF task, which runs completely independently, and is not subservient to any other ELF task.

For example, if you run a macro from a spreadsheet cell, the macro is terminated when you exit the spreadsheet. However, if your macro contains the line DB_OWNERLESS@(dbox, TRUE), the macro continues to run when you exit the Spreadsheet.

DB_PAINT@

Draws a dialog box on the screen

Format DB_PAINT@(dbox)

Arguments dbox The name of the dialog box variable.

Description Paints a dialog box. That is, this macro displays the dialog box but does not make it ready to accept commands. Use this macro if you must read state information from the dialog box and then change the dialog box based on this state information.

This macro is most often used when you want to initialize information in an edit box or in a table.

Example

DB_RECT_STYLE@

Sets the style of the Tab Widget

Format DB_CTRL_RECT_STYLE@(format dialog_box_ dbox, ID, style)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the control.
Style	0 = Plain
	1 = Labeled
	2 = layered

Description Sets the style of the tab control in a dialog box. A labeled tab control has a string at the top that can be used for descriptive purposes. A layered tab control allows you to group related controls in separate tabs.

See also DB_TABCTRL_SET_LAYERNAMES@

DB_REFRESH@

Redisplays a dialog box

Format DB_REFRESH@(dbox)

Arguments

dbox	The name of the dialog box variable.

Description Redisplays an up and running dialog box. While DB_DISPLAY@ will do the same thing, this macro is faster because it assumes that the state of the dialog box has not changed since the last time it was displayed.

See also DB_DISPLAY@

DB_RESIZE_INFO@

Returns the new size of a dialog box

Format string = DB_RESIZE_INFO@(dbox)

Arguments

dbox	The name of the dialog box variable.

Description Returns a string containing the new width and height of a dialog box after its window has been resized. The format of the returned string:

new_width:new_height

newWidth and newHeight are numbers representing the new width and height of the dialog box in pixels. The new dimensions may then be used to reposition and resize dialog box controls and labels.

A dialog box window is made resizeable using DB_VAR_SIZE@. When a dialog box window is resized, the resize_ exit condition is triggered. DB_RESIZE_INFO@ should be used only after determining that the exit code is resize_.

See also DB_VAR_SIZE@

DB_WIDTH@

DB_HEIGHT@

DB_SEND_POKE@

Sends a poke message

Format DB_SEND_POKE@(code[, message])

Arguments

code	A unique number assigned to the poke message. code must be less than 10,000.
message	The message to send. This value can be any ELF data type including a multi-level array.

Description Sends the message you specify. The message will be received by any tasks that include a DB_ACCEPT_POKES@ macro for the message.

See also DB_ACCEPT_POKES@

DB_GET_POKE@

DB_GET_POKE_DATA@

DB_SUSPEND@

Suspends DB_DISPLAY@ action after a callback is executed

Format DB_SUSPEND@( )

Description Suspends execution so that additional processing can occur. Normally, action is only suspended after a DB_DISPLAY@ executes. This defines a second point at which you can process your callback functions.

This sets the *suspend* state for all callback functions within the dialog box. That is, you cannot suspend interactions so that only one callback function is invoked.

DB_TABCTRL_ACTIVE_LAYER@

Sets the active layer in a tab control

Format DB_TABCTRL_ACTIVE_LAYER@(dbox, ID, layerName)

Arguments

dbox	The name of the dialog box variable.
ID	The ID of the tab control.
layerName	The name of the layer to make active.

Description Sets the active layer in a tab control. That layer, and all the controls on that layer, are moved to the top of the display, and all other layers are moved back one layer.

See also DB_TABCTRL_GET_ACTIVE_LAYER@

DB_TABCTRL_GET_ACTIVE_LAYER@

Returns the name
of the active layer in a tab control

Format DB_TABCTRL_GET_ACTIVE_LAYER@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The ID of the tab control.

Description Returns a string containing the name of the currently-active layer in a tab control.

See also DB_TABCTRL_ACTIVE_LAYER@

DB_TABCTRL_INSERT_CONTROL@

Adds a control to a
layer in a tab widget

Format DB_TABCTRL_INSERT_CONTROL@(dbox, ID, layerName, newControlId)

Arguments

dbox	The name of the dialog box variable
ID	The ID of the tab control
layerName	The layer to who which the new control is added
newControlId	The control ID of the new widget added to layerName

Description Adds a control to a layer in a tab control. The control added must have been created with the DB_CREATE_CTRL@ macro.

See also DB_TABCTRL_ACTIVE_LAYER@, DB_TABCTRL_GET_ACTIVE_LAYER@, DB_CREATE_CTRL@

DB_TABCTRL_SET_LAYERNAMES@

Establishes the Tab
names for a layered panel

Format DB_TABCTRL_SET_LAYERNAMES@(dbox, ID, layerNames)

Arguments

dbox	The name of the dialog box variable
ID	The ID of the layered panel control
layernames	an array of string names that appear on the tabs of the layered panel.

Description Establishes the names that display on the tabs of a layered panel.

DB_TABLE_ALLOW_COLUMN_RESIZING@

Allows interactive column resizing

Format DB_TABLE_ALLOW_COLUMN_RESIZING@(dbox, ID, flag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the table.

flag	If flag is TRUE, resizing is allowed. If FALSE, column size cannot change.

Description Allows a user to change (or prevents a user from changing) the size of a table's columns.

DB_TABLE_ALLOW_EDITING@

Allows editing of table cells

Format DB_TABLE_ALLOW_EDITING@(dbox, ID, flag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the table.
flag	If flag is TRUE, editing of cells is allowed. If flag is FALSE, editing is not allowed. The default is FALSE.

Description Allows a user to edit (or prevents a user from editing) the contents of a table's cells.

DB_TABLE_ALLOW_HIDDEN_COLUMNS@

Indicates if columns within a table can be hidden

Format DB_TABLE_ALLOW_HIDDEN_COLUMNS@(dbox, ID, flag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the table.
flag	If flag is TRUE, hiding columns is allowed. If flag is FALSE, hiding is not allowed. The default is FALSE.

Description Allows a user to hide one or more of a table's columns. While this macro allows the user to hide columns, it does not hide them.

DB_TABLE_CLEAR_DATA@

Clears rows of data from a table

Format DB_TABLE_CLEAR_DATA@(dbox, ID, startRow, numRows[, repaintFlag])

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the table.

	startRow	The first row that will be cleared.
	numRows	The number of rows to be cleared. If numRows is set to -1, all rows are cleared.
	repaintFlag	An optional value that indicates that the entire table will be redrawn rather than just the rows that were cleared

Description Clears numRows rows of data beginning at startRow.

See also DB_TABLE_GET_DATA@
DB_TABLE_SET_DATA@

DB_TABLE_EFFICIENT_SCROLL@

Turns on and off efficient scrolling

Format DB_TABLE_EFFICIENT_SCROLL@(dbox, ID, flag)

Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the table.
	flag	If flag is TRUE, efficient type scrolling is used. If flag is FALSE, the default scrolling is used. The default is FALSE.

Description Enables and disables the "efficient" scrolling method. Efficient scrolling keeps your cursor centered in the table while you are scrolling up or down. It actually performs a half page scroll when the cursor hits the bottom of the displayed portion of your table. Turn this function on to increase scrolling performance on slow terminals.

DB_TABLE_GET_DATA@

Retrieves table information

Format stringArray = DB_TABLE_GET_DATA@(dbox, ID)

Arguments	dbox	The name of the dialog box variable.
Arguments	ID	The string name of the table.

Description Retrieves the information contained within a table. This information is returned as an array of strings.

See also DB_TABLE_CLEAR_DATA@

DB_TABLE_SET_DATA@

DB_TABLE_GET_DISP_LINES@

Returns the table's row height

Format rowHeight = DB_TABLE_GET_DISP_LINES@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the table.

Description Returns the number of display lines in a table. That is, this is the height of the table expressed in rows.

DB_TABLE_GET_ROWS@

Retrieves row data

Format stringArray = DB_TABLE_GET_ROWS@(dbox, ID, startRow, count)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the table.
startRow	The first row from which data will be retrieved.
count	The number of rows of data to be retrieved.

Description Retrieves the data from one or more table rows.

DB_TABLE_GET_SELECTIONS@

Retrieves current selections in a table

Format intArray = DB_TABLE_GET_SELECTIONS@(dbox, ID)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the table.

Description Retrieves an array containing the row number of selected rows. An array of zero-based row numbers is returned.

See also DB_TABLE_SET_SELECTIONS@

DB_TABLE_GET_TOP_ROW@

Returns the top row number being displayed

Format rowNum = DB_TABLE_GET_TOP_ROW@(dbox, ID)

Arguments		
	dbox	The name of the dialog box variable.
	ID	The string name of the table.

Description Returns the row number of the top row being displayed. This is a 0-based number.

See also DB_TABLE_SET_NEW_TOP_ROW@

DB_TABLE_GOTO_CELL@

Places the cursor at a specified cell in the table

Format DB_TABLE_GOTO_CELL@(dbox, ID, row, col[, startChar])

Arguments		
	dbox	The name of the dialog box variable.
	ID	The string name of the table.
	row	A row within the table.
	col	A column within the table.
	startChar	A number defining the place where the cursor will be placed within the cell. If omitted, the cursor is placed before the first character.

Description Places the cursor before the first character in a cell.

If the value of row and col are both -1, the data within the table is taken from the displayed table and placed within the table's data structure; that is, these values indicate that the values typed into the table's cells should be committed.

DB_TABLE_INSERT_TEXT@

Inserts text at cursor

Format DB_TABLE_INSERT_TEXT@(dbox, ID, text)

Arguments		
	dbox	The name of the dialog box variable.

	ID	The string name of the control.
	text	The text you want to insert.

Description Inserts text at the current cursor location in the table.

DB_TABLE_MARKER_PIXMAPS@

Sets marker pixmaps at a starting row

Format DB_TABLE_MARKER_PIXMAPS@(dbox, ID, startRow, markerPixmaps)

Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the table.
	startRow	The starting row number.
	markerPixmaps	The array of pixmaps to use within the table.

Description Sets marker pixmaps starting from row startRow. If the markerPixmaps argument is set to NULL, the macro deletes all pixmaps beginning at startRow.

DB_TABLE_NEXT_PAGE@

Moves to the next page

Format DB_TABLE_NEXT_PAGE@(dbox, ID)

Arguments	dbox	The name of the dialog box variable.
	ID	The string name of the table.

Description Displays the *next* page in a table.

See also DB_TABLE_PREVIOUS_PAGE@

DB_TABLE_PREVIOUS_PAGE@

Moves to the previous page

Format DB_TABLE_PREVIOUS_PAGE@(dbox, ID)

Arguments	dbox	The name of the dialog box variable.

ID	The string name of the table.

Description Displays the *previous* page in a table.

See also DB_TABLE_NEXT_PAGE@

DB_TABLE_ROW_IS_KNOWN@

Indicates if a row exists

Format flag = DB_TABLE_ROW_IS_KNOWN@(dbox, ID, row)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the table.
row	The row number of a row within the table.

Description Returns a Boolean value indicating if the row exists within a table. TRUE indicates that the row exists.

DB_TABLE_SET_DATA@

Replaces a table's contents

Format DB_TABLE_SET_DATA@(dbox, ID, rows, headings, markerPixmaps)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the table.
rows	A two-dimensional array of row information.
headings	The headings for the table's columns. This argument is a 2-dimensional array or an array of arrays where the first element of the sub-array is the header string and the second element is the width of the field in pixels.
markerPixmaps	A two-dimensional array of marker pixmap information.

Description Replaces a table's contents.

Example

See also DB_TABLE_CLEAR_DATA@
DB_TABLE_GET_DATA@

DB_TABLE_SET_NEW_DATA@

DB_TABLE_SET_FONT@

Sets a table's fonts

Format DB_TABLE_SET_FONT@(dbox, ID, fontFamily, pointSize, boldFlag, italicFlag)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the table.
fontFamily	The font family of the table.
pointSize	The font's point size.
boldFlag	A Boolean value which if set to TRUE indicates that the text will be displayed using a bold font.
italicFlag	A Boolean value which if set to TRUE indicates that the text will displayed using an italic font.

Description Sets a table's font and point size.

DB_TABLE_SET_HSCROLL@

Sets the table's horizontal scroll bar

Format DB_TABLE_SET_HSCROLL@(dbox, ID, origin, length)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the table.
origin	The origin of the horizontal scroll bar in pixels.
length	The length of the horizontal scroll bar in pixels.

Description Sets a table's horizontal scroll bar's origin and length.

See also DB_TABLE_SET_VSCROLL@

DB_TABLE_SET_MARKER_WIDTH@

Sets the table's marker width

Format DB_TABLE_SET_MARKER_WIDTH@(dbox, ID, npix)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the table.
npix	The width of the marker in pixels.

Description Sets the row marker's width in pixels for the table.

DB_TABLE_SET_NEW_DATA@

Replaces a table's contents

Format DB_TABLE_SET_NEW_DATA@(dbox, ID, newData, startRow)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the table.
newData	The new information that will replace information that currently exists in the table.
startRow	The starting row.

Description Replaces a table's contents beginning at startRow with newData.

See also DB_TABLE_SET_DATA@

DB_TABLE_SET_NEW_TOP_ROW@

Sets a table's new top row

Format DB_TABLE_SET_NEW_TOP_ROW@(dbox, ID, newTopRow)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the table.
newTopRow	A number designating the new top row.

Description Sets the new top row for a table. The table will manage the scrollbar as needed.

Example

See also DB_TABLE_GET_TOP_ROW@

DB_TABLE_SET_SELECTIONS@

Selects rows

Format DB_TABLE_SET_SELECTIONS@(dbox, ID, selections)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the table.
selections	An array of zero-based row numbers.

Description Sets selections in a table. That is, this selects rows in a table. To clear all selections in a table, pass NULL as the selections argument.

See also DB_TABLE_GET_SELECTIONS@

DB_TABLE_SET_VSCROLL@

Sets the table's vertical scroll bar's origin and length

Format DB_TABLE_SET_VSCROLL@(dbox, ID, origin, length)

Arguments

dbox	The name of the dialog box variable.
ID	The string name of the table.
origin	The origin of the vertical scroll bar in pixels.
length	The length of the vertical scroll bar in pixels.

Description Sets the table's vertical scroll bar's origin and length.

See also DB_TABLE_EFFICIENT_SCROLL@
DB_TABLE_SET_HSCROLL@

DB_TIMER@

Sets "timeout" for a dialog box

Format DB_TIMER@(dbox,seconds)

Arguments

dbox	The name of the dialog box variable.
seconds	The number of seconds for "timeout".

Description Sets the number of seconds that will occur before **DB_DISPLAY@** times out and returns. If no events have occurred to the dialog box in this time, a timer_ exit condition occurs.

The default value of seconds is infinity. That is, control is not returned to the dialog box until an event occurs.

Example

DB_TITLE@

Sets a dialog box's title

Format DB_TITLE@(dbox, title)

Arguments

dbox	The name of the dialog box variable.
title	The string to be displayed in the dialog box title area.

Description Sets a dialog box's title. Normally, titles are set using the Dialog Box Editor. However, you can use DB_TITLE@ to set or change a dialog box title.

DB_VAR_SIZE@

Allows a dialog box window to be resized

Format DB_VAR_SIZE@(dbox, [minWid, minHt, [maxWid[, maxHt]]])

Arguments

dbox	The name of the dialog box variable.
minWid	The minimum width of the dialog box window, in pixels.
minHt	The minimum height of the dialog box window, in pixels.

	maxWid	The maximum width of the dialog box window, in pixels.
	maxHt	The maximum height of the dialog box window, in pixels.

Description Allows a dialog box window to be resized using a mechanism provided by the window manager (usually by dragging a window corner with the mouse pointer). Any macro using DB_VAR_SIZE@ should be prepared to reposition and resize dialog box controls and labels based on the new size.

When a dialog box window is resized, the exit condition resize_ is triggered. DB_RESIZE_INFO@ can then be used to determine the new size of the dialog box. The size limits in the argument list are the minimum and maximum sizes of the *window* containing the dialog box, and not of the dialog box itself. The height of the window is the height of the dialog box, plus the heights of the Menu Bar and *Express*Line (if they exist).

See also DB_RESIZE_INFO@

DB_WIDTH@

DB_HEIGHT@

DB_VIEW_EXPRESSLINE@

Turns on the *Express*Line

Format DB_VIEW_EXPRESSLINE@(dbox)

Arguments dbox The name of the dialog box variable.

Description Turns on or off viewing of the *Express*Line.

DB_WIDTH@

Sets the width of a dialog box

Format DB_WIDTH@(dbox, width)

Arguments	dbox	The name of the dialog box variable.
	value	The width, in pixels, of the dialog box.

Description Normally, the dialog box width is set using the Dialog Box Editor. If you set the width using DB_WIDTH@, make sure that the dialog box can contain all the controls in the dialog box.

See also DB_GET_WIDTH@
DB_HEIGHT@

DB_WINDOW_REMAIN@

Determines how a dialog box display is redisplayed

Format DB_WINDOW_REMAIN@(dbox, flag)

Arguments

dbox	The name of the dialog box variable.
flag	If TRUE, the dialog box display is maintained when a push button of type Execute & Dismiss is pressed or the dialog box is redisplayed using DB_DISPLAY@. If FALSE, the dialog box is not maintained.

Description Indicates if a dialog box display is maintained when an Execute & Dismiss push button is pressed or the dialog box is redisplayed using DB_DISPLAY@.

You would use this macro when you want a button of type Execute & Dismiss to perform an action but not necessarily exit the dialog box or when you want to redisplay a dialog box without first having suspended dialog box operation. DB_WINDOW_-REMAIN@ has no effect on buttons of type Dismiss, Help, Bitmap, Execute, or Normal.

Example

DB_XLATE_FONT@

Displays the name of the xfont being used

Format font = DB_XLATE_FONT@(infont, axToXflag[, anyFlag])

Arguments

infont	The font currently being used.
axToXflag	If TRUE returns the X font name of infont; otherwise, the name of the font being used is displayed.
anyFlag	A Boolean value which if set to TRUE indicates that both PostScript and PCL5 fonts are handled. If FALSE, only the current font family is handled; a font from the "other" family is not recognized. The default is FALSE.

Description Translates between the X dialog box font and the matching Applix*ware* font. The Times-Roman font is returned if the passed font is not recongized. If axToXflag is set to TRUE, infont should be an Applix*ware* font. Otherwise, infont should be an X font.

Example

DB_XPOS@

Specifies the dialog box's X-position

Format DB_XPOS@(dbox, value)

Arguments

dbox	The name of the dialog box variable.
value	A number or number array indicating the x-axis location, in pixels, of the top left corner of the dialog box. If values for both DB_XPOS@ and DB_YPOS@ are 0, the dialog box is centered under the current mouse pointer location.

Description Normally, when a dialog box is displayed it is centered under the current mouse pointer location. You can use DB_XPOS@ and **DB_YPOS@** to indicate a specific location where the dialog box should be displayed. DB_XPOS@ must appear before **DB_DISPLAY@**.

Since values of 0 for both DB_XPOS@ and DB_YPOS@ indicate that the dialog box be displayed under the current mouse pointer position, the closest that you can position a dialog box to the top left of a screen is position (1,0) or position (0,1).

If you use DB_XPOS@ to set the horizontal position of the dialog box without using DB_YPOS@ to set the vertical position, then the vertical position is set to 0. The maximum values for DB_XPOS@ and DB_YPOS@ will depend on the dimensions of your screen.

DB_YPOS@

Specifies the dialog box's Y-position

Format DB_YPOS@(dbox, value)

Arguments

dbox	The name of the dialog box variable.
value	A number or number array indicating the y-axis location, in pixels, of the top left corner of the dialog box. If values for both DB_XPOS@ and DB_YPOS@ are 0, the dialog box is centered under the current mouse pointer location.

Description Normally, when a dialog box is displayed it is centered under the current mouse pointer location. You can use **DB_XPOS@** and DB_YPOS@ to indicate a specific location where the dialog box should be displayed. DB_YPOS@ must appear before DB_DISPLAY@.

Since values of 0 for both DB_XPOS@ and DB_YPOS@ indicate that the dialog box be displayed under the current mouse pointer position, the closest that you can position a dialog box to the top left of a screen is position (1,0) or position (0,1).

If you use DB_YPOS@ to set the vertical position of the dialog box without using DB_XPOS@ to set the horizontal position, then the horizontal position is set to 0. The maximum values for DB_XPOS@ and DB_YPOS@ will depend on the dimensions of your screen.

DD_DIRECTORY_DISPLAY@

Displays the Directory Displayer

Format DD_DIRECTORY_DISPLAY@()

Description Starts the Directory Displayer which displays all Applix*ware* and non-Applix*ware* files and directories from the current directory.

NOTE: If more than 1,000 files exist within a directory, DD_DIRECTORY_DISPLAY@ will fail. You can correct this problem by setting the following three Directory Displayer preference options:

Maximum Number of Files to Search through

Maximum Levels in a Recursive Search

Maximum Search Time in Seconds

DECOMPOSE_TIME@

Changes a date number to an array

Format timeArray = DECOMPOSE_TIME@(timeValue)

Arguments timeValue A date/time value.

Description Changes a date number to an array. The array is of format data_time_array_ as defined in the ELF include file datetim_.am. In most cases, timeValue is the value returned by CURRENT_TIME@. The elements of the format date_time_array_ are as follows:

· Seconds:	0 - 59

· Minutes:	0 - 59
· Hour:	0 - 23
· Day:	0 - 30
· Month:	0 - 11
· Year:	(year - 1900)
· Weekday:	0 - 6, Sunday is 0
· Yearday:	0 - 365
· Is daylight savings time:	1 if TRUE

Note that several of these ranges are zero-based. Therefore, if the date is 9/11/95, 8 is returned for the month, 10 is returned for the day, and so on.

Example

DEFINE_ERROR_MESSAGE@

Defines an error string

Format DEFINE_ERROR_MESSAGE@(number, string)

Arguments

number	The error message number.
string	An error message. The maximum length of this string is 1024 characters.

Description Defines an error string and associates it with an error number.

DEFINE_KEY@

Adds a keyboard key to the list of recognized keys

Format DEFINE_KEY@(name, value)

Arguments

name	The name of the key to add to the recognized list of keys. name can be any key listed in the /usr/include/X11/keysym.h file.
value	A numeric value to associate with the key specified by name. Whenever the key specified by name is pressed, it will produce the keysym value specified by value.

Description The following table listing the keyboard keys recognized by ELF. The table lists the key name as specified in ELF and the X-window key name as specified in the file /usr-/include/X11/keysym.h. The keys recognized by ELF are a subset of all the keys available as listed in keysym.h.

Some keyboards use engraved keypad names that do not match the names in the following list. In such event, refer to that keyboard's reference manual for the X-windows key name.

Key Name	**X-windows key id**
BackSpace	XK_BackSpace
Tab	XK_Tab
Linefeed	XK_Linefeed
Clear	XK_Clear
Return	XK_Return
Pause	XK_Pause
Escape	XK_Escape
Delete	XK_Delete
F1	XK_F1
F2	XK_F2
F3	XK_F3
F4	XK_F4
F5	XK_F5
F6	XK_F6
F7	XK_F7
F8	XK_F8
F9	XK_F9
F10	XK_F10
F11	XK_F11
F12	XK_F12
F13	XK_F13
F14	XK_F14
F15	XK_F15
F16	XK_F16
F17	XK_F17
F18	XK_F18
F19	XK_F19
F20	XK_F20
F21	XK_F21
F22	XK_F22
F23	XK_F23
F24	XK_F24
F25	XK_F25
F26	XK_F26

F27	XK_F27	
F28	XK_F28	
F29	XK_F29	
F30	XK_F30	
F31	XK_F31	
F32	XK_F32	
F33	XK_F33	
F34	XK_F34	
F35	XK_F35	
Home	XK_Home	
Left	XK_Left	
Up	XK_Up	
Right	XK_Right	
Down	XK_Down	
Prev	XK_Prior	
Prior	XK_Prior	
Next	XK_Next	
End	XK_End	
Begin	XK_Begin	
Select	XK_Select	
Print	XK_Print	
Execute	XK_Execute	
Insert	XK_Insert	
Undo	XK_Undo	
Redo	XK_Redo	
Menu	XK_Menu	
Find	XK_Find	
Cancel	XK_Cancel	
Help	XK_Help	
Break	XK_Break	
KP_Space	XK_KP_Space	(numeric key pad)
KP_Tab	XK_KP_Tab	(numeric key pad)
KP_Enter	XK_KP_Enter	(numeric key pad)
PF1	XK_KP_F1	(numeric key pad)
PF2	XK_KP_F2	(numeric key pad)
PF3	XK_KP_F3	(numeric key pad)
PF4	XK_KP_F4	(numeric key pad)
KP_Equal	XK_KP_Equal	(numeric key pad)
KP_Multiply	XK_KP_Multiply	(numeric key pad)
KP_Add	XK_KP_Add	(numeric key pad)
KP_Separator	XK_KP_Separator	(numeric key pad)
KP_Subtract	XK_KP_Subtract	(numeric key pad)
KP_Decimal	XK_KP_Decimal	(numeric key pad)
KP_Divide	XK_KP_Divide	(numeric key pad)

KP_0	XK_KP_0	(numeric key pad)
KP_1	XK_KP_1	(numeric key pad)
KP_2	XK_KP_2	(numeric key pad)
KP_3	XK_KP_3	(numeric key pad)
KP_4	XK_KP_4	(numeric key pad)
KP_5	XK_KP_5	(numeric key pad)
KP_6	XK_KP_6	(numeric key pad)
KP_7	XK_KP_7	(numeric key pad)
KP_8	XK_KP_8	(numeric key pad)
KP_9	XK_KP_9	(numeric key pad)
Remove	DXK_Remove	(DEC only)
Space	XK_space	
A	XK_a	
B	XK_b	
C	XK_c	
D	XK_d	
E	XK_e	
F	XK_f	
G	XK_g	
H	XK_h	
I	XK_i	
J	XK_j	
K	XK_k	
L	XK_l	
M	XK_m	
N	XK_n	
O	XK_o	
P	XK_p	
Q	XK_q	
R	XK_r	
S	XK_s	
T	XK_t	
U	XK_u	
V	XK_v	
W	XK_w	
X	XK_x	
Y	XK_y	
Z	XK_z	
0	XK_0	
1	XK_1	
2	XK_2	
3	XK_3	
4	XK_4	
5	XK_5	

6	XK_6
7	XK_7
8	XK_8
9	XK_9
!	XK_exclam
"	XK_quotedbl
#	XK_numbersign
$	XK_dollar
%	XK_percent
&	XK_ampersand
"	XK_quoteright
(	XK_parenleft
)	XK_parenright
*	XK_asterisk
+	XK_plus
,	XK_comma
-	XK_minus
.	XK_period
/	XK_slash
:	XK_colon
;	XK_semicolon
<	XK_less
=	XK_equal
>	XK_greater
?	XK_question
@	XK_at
[	XK_bracketleft
\\	XK_backslash
]	XK_bracketright
^	XK_asciicircum
_	XK_underscore
'	XK_quoteleft
{	XK_braceleft
|	XK_bar
}	XK_braceright
~	XK_asciitilde

When you define a key using DEFINE_KEY@, it is added to the list of keys recognized by ELF. Any key recognized by ELF can be used as an accelerator key for menu options.

DE_CHANGE_DIALOG@

Change dialog command

Format DE_CHANGE_DIALOG@()

Description Invokes the Dialog Box editor so that you can change an existing dialog box.

See also **DE_EDIT_DIALOG@**

DE_EDIT_DIALOG@

Invokes the dialog box editor

Format DE_EDIT_DIALOG@([name])

Arguments	name	The name of an existing dialog box file.

Description Invokes the dialog box editor. If you include the optional name argument, that file is loaded into the dialog box editor. Otherwise, the dialog box editor is brought up in its initial state so that you can create a new dialog box.

See also **DE_CHANGE_DIALOG@**

DELAY@

Delays macro execution

Format DELAY@(seconds)

Arguments	seconds	A number indicating the number of seconds to delay execution of a macro.

Description Delays the execution of a macro for a specified number of seconds. After the specified number of seconds elapses, the macro resumes execution at the statement following DELAY@.

When you are running an application that uses ELF tasks, DELAY@ is used to pass control of the execution thread to the ELF scheduler. See the discussion of the **ELF scheduler** for more information.

In addition, the statement DELAY@(0) guarantees that your macro will be allowed one more chance to run by the ELF scheduler. An ELF macro task which would ordinarily be terminated by the scheduler, such as a macro whose macro parent task has exited, is not destroyed if it is in a sleep condition caused by DELAY@(0). This task would be destroyed if it called DELAY@(1) instead.

Example

DELETE_DIRECTORY@

Deletes a directory

Format DELETE_DIRECTORY@(dir)

Arguments

dir	The path of the directory to be deleted.

Description Deletes an empty directory. This macro generates an error if you try to delete a directory that is not empty.

See also DELETE_FILE@
DELETE_FILE_RECURSIVE@

DELETE_FILE@

Deletes a file

Format DELETE_FILE@(file)

Arguments

file	The name of the file you want to delete. You can specify a path name if desired. file must include the filename extension.

Description Removes a file from the file system. Use this macro with caution. A file deleted using DELETE_FILE@ cannot be retrieved. DELETE_FILE@ does not generate an error if it cannot delete the specified file. DELETE_FILE@ cannot delete directories.

See also DELETE_DIRECTORY@
DELETE_FILE_RECURSIVE@

DELETE_FILE_RECURSIVE@

Deletes a directory and its contents

Format DELETE_FILE_RECURSIVE@(dir)

Arguments dir The path name of the directory to be deleted.

Description The equivalent of the UNIX command rm -rf. Use this macro with caution. Directories and files deleted using DELETE_FILE_RECURSIVE@ cannot be retrieved. DELETE_FILE_RECURSIVE@ does not generate an error if it cannot delete dir.

See also DELETE_FILE@
DELETE_DIRECTORY@

DEMOS_DIR@

Returns the pathname of the demos directory

Format DEMOS_DIR@()

Description The Demos directory contains sample ELF programs and other applications. Some of these are available from the Applix*ware* main menu when you select Utilities ® Sample Macros. The default Demos directory is *install_dir*/axdata/*lang*/Demos, where *install_dir* is the installation path of your software and *lang* is the language directory.

DESTROY_GRAPHIC@

Destroys the graphic instance

Format DESTROY_GRAPHIC@(gfx)

Arguments gfx A graphics handle.

Description Frees up all resources associated with the gfx graphics handle. After executing this macro, the graphic object can no longer be manipulated.

See also ASSIGN_GRAPHIC@
CREATE_GRAPHIC@
GET_GRAPHIC@

LOAD_GRAPHIC@
READ_GRAPHIC_FILE@

DIALOG_LIST@

Returns the names of open dialog boxes

Format nameArray = DIALOG_LIST@()

Description Returns a list containing the names of all open dialog boxes.

DIR_EXISTS@

Determines if a directory exists

Format flag = DIR_EXISTS@(name)

Arguments name A string indicating the directory to search for.

Description Returns TRUE if a directory with the name specified by name exists. Returns FALSE if a directory having that name does not exist.

Example

DIR_SLASH@

Returns the slash directory separator for operating system

Format for_or_back_slash = DIR_SLASH@( )

Description Returns the *slash* character used by the operating system to separate directory/-sub-directory information. If you are using UNIX, a forward slash ("/") is returned. If you are using DOS, a backward slash ("\") is returned.

DOC_HAS_LINKS@

Indicates if a document has links

Format flag = DOC_HAS_LINKS@(filename)

Arguments filename The name of a file being tested for links.

Description Returns a TRUE/FALSE value indicating if document filename contains any links.

DONE_WITH_FILE@

Deletes a file from the internal list of open files

Format DONE_WITH_FILE@(filename)

Arguments filename The file being removed from the Applix*ware* internal usage list.

Description Removes *filename* from Applix*ware*'s internal list of open files. No other action occurs. That is, while you are telling Applix*ware* that the file is no longer be used, it can still be up in a window. The implications of this action are:

- You can now open the file in another window; that is, the same information will be in two windows. However, no safeguards exist to prevent edits in one window from overwriting edits made in the other. That is, the operations performed in one window are independent of the operations performed in the other.
- The value returned by the **IS_FILE_OPEN@** macro will be FALSE, even though the file is open.

DUMP_ARRAY@

Displays a window that lists all elements of an array and its sub-array

Format DUMP_ARRAY@(array[, maxDepth])

Arguments	array	The array whose elements are to be listed. array can contain strings, numbers, or arrays.
	maxDepth	Indicates how many array levels will be displayed when the DUMP_ARRAY@ window first appears. If this argument is omitted, the default is 1. (If you set this value to 0, 1 level is displayed.)

Description Invokes a window that lists all elements of a given array and its sub-arrays. This is a useful debugging tool that lets you examine data that has been assigned to variables before acting on the data. This window suspends operation of its parent macro until the window is dismissed.

Example

EDATE@

Returns the last day of the month
that falls a given number of months before or after a start date

Format EDATE@(startDate, months)

Arguments

startDate	a string containing a formatted date, such as "May 16, 1993" or "5/16/93"
months	A numeric value indicating the number of months before or after the start-Date. This value can be negative. Negative numbers are subtracted from the start date.

Description EDATE@ calculates the exact date that falls a given number of months before or after the start_date.

The startDate argument is a number representing the date. This is the number of days the given date falls after 12/31/1899. This number is returned by the macro DAT-EVALUE@.

If months is a positive integer, the EDATE@ function returns a date after the start start date; if months is negative, the function returns a date before the start date.

EDATE@(DATEVALUE@("12/15/95"),1) equals 35079 or 01/15/96

EDATE@(DATEVALUE@("12/31/95"),-1) equals 35033 or 11/30/95

EDITPREFS@

Invokes the Preferences Editor dialog box

Format EDITPREFS@()

Description Displays the Preferences Editor dialog box, which provides an easy method of customizing Applix*ware*. If you change any setting in this dialog box, **CHANGE_PREFS@** reacts by editing the user's ax_prof4 file accordingly.

Applix*ware* makes considerable use of preferences. This dialog box lets you edit many of these preferences. Other preferences are not designed to be changed by users. If you ever have need to change any of this category of preference, change them using any ASCII text editor.

For more information, see **Applixware Preferences Editor**.

EDIT_ICON@

Starts the Applixware Bitmap Editor

Format EDIT_ICON@(filename[, width, height])

Arguments

filename	A string indicating the name of the file to open or create, with the .im extension.
width	A number indicating the width to make the new bitmap image. Do not include this argument if loading an existing image file.
height	A number indicating the height to make the new bitmap image. Do not include this argument if loading an existing image file.

Description If the file specified by filename exists, it is loaded in the Bitmap Editor. If the file does not exist, it is created and the Bitmap Editor is started.

ELF_COMPILE_ARRAY@

Compiles an array of data

Format binary = ELF_COMPILE_ARRAY@(moduleName, stringsToCompile)

Arguments

moduleName	The name of the file containing the strings. This name is stored so that the file can be accessed by the debugger and other systems that need to know a function's moduleName. This name should differ from the name of the functions being compiled.
stringsToCompile	The information being compiled.

Description Compiles an array of strings, returning the compiled program. If binary is not an array, it contains error strings.

Typically, this macro is used in the following way:

```
    code = READ_ASCII_FILE@(``/home/mycode.ascii")
    stuff = ELF_COMPILE_ARRAY@ (``/home/mycode.ascii", code)
```

followed by either:

```
    WRITE_DATA_FILE@(``/release/to_all/mycode.elo", stuff)
or
    ELF_INSTALL_BINARY@(stuff)
```

See also ELF_INSTALL_BINARY@

ELF_GET_CLIPBOARD@

Returns the contents of the clipboard

Format stringArray = ELF_GET_CLIPBOARD@()

Description Returns a string array listing the contents of the clipboard in any Applix*ware* window. The clipboard is filled using one of the following macros:

- DB_CTRL_COPY@
- DB_CTRL_CUT@
- ELF_SET_CLIPBOARD@
- GR_COPY@
- GR_CUT@
- ME_COPY@
- ME_CUT@
- SS_COPY@
- SS_CUT@
- WP_COPY@
- WP_CUT@

The returned stringArray can be multidimensional. For example, the returned array from a marked area in Spreadsheets would be a 2-dimensional array of rows and columns

See also ELF_GET_SELECTION@

ELF_GET_SELECTION@

Returns the window system's current selection

Format stringArray = ELF_GET_SELECTION@()

Description Returns a string array containing the window system's current selection picked with the mouse. See also ELF_GET_CLIPBOARD@, which returns the contents of the *clipboard.*

The returned stringArray can be multidimensional. For example, the returned array from a marked area in Spreadsheets would be a 2-dimensional array of rows and columns

ELF_INSTALL_BINARY@

Loads a binary ELF program

Format ELF_INSTALL_BINARY@(compiledData)

Arguments

compiledData	A compiled function or set of functions. This data is returned by ELF_COMPILE_ARRAY@.

Description Loads an ELF program file containing binary information. If the file is already loaded, it is unloaded, then reloaded. The functions in this file are loaded at the beginning of ELF's function list. In this way, these functions are called before functions already loaded.

ELF_MESSAGE@

Sends a message to another ELF task

Format retMsg = ELF_MESSAGE@(taskID, cmdCode, data)

Arguments

taskID	The ID of the task to which you are sending data.
cmdCode	An arbitrary command code that will be understood by task taskID. In most cases, this value is a number stored in a header file included by both tasks.
data	The data being sent. This can be a single-element variable or an array.

Description Sends a message to another ELF task. The task identified by taskID must be running and ready to receive messages (using GET_MESSAGE@).

Because the task to which you are sending this message must return data to this macro, the receiving macro must know the task ID of the task executing the ELF_MESSAGE@ macro. Typically, this is done by placing the current task's ID as the first element of data. For example, if you were sending a string to a task, you would send a two-element array:

· The first element contains the originating task's task I.

· The second element contains the string.

A response must be returned from that task. (The task will send the response using ELF_RESPOND@ or ELF_RESPOND_ERROR@).

ELF_PARENT_TASK@

Returns the parent task's process ID

Format ID = ELF_PARENT_TASK@()

Description Returns the ID of the current task's parent. (That is, it is the task id of the task from which the current task originated.)

See also ELF_TASK_ID@

MACRO_PARENT_TASK@

ELF_RESPOND@

Respond to an ELF_MESSAGE@

Format ELF_RESPOND@(taskID, data)

Arguments

taskID	The task ID of the task that sent a message.
data	The message being sent back to the task that sent a message.

Description Sends data back to the task that sent the current task a message. (This message was received using GET_MESSAGE@.) Typically, the sending task places it task ID in the first element of the data it sends to this task. However, other methods such as global variables can also be used.

See Also GET_MESSAGE@

ELF_MESSAGE@
ELF_RESPOND_ERROR@

ELF_RESPOND_ERROR@

Throws an error to a task

Format ELF_RESPOND_ERROR@(taskID, code, errorString[, object])

Arguments

taskID	The task ID of the task that sent a message.
code	A unique error number. code must be in the range from 1 to 1000.
errorString	The error string that is displayed in an error dialog box.
object	A string representing the object of an error. This argument is optional.

Description Throws an error to the task identified with taskID.

See Also GET_MESSAGE@
ELF_MESSAGE@
ELF_RESPOND@

ELF_SET_CLIPBOARD@

Places a string array in the clipboard

Format ELF_SET_CLIPBOARD@(stringArray)

Arguments

stringArray	A string array of text to be copied to the clipboard.

Description Places an array of text strings in the clipboard. The clipboard can be accessed in any Applix*ware* window, with any of the following macros:

- ME_PASTE@
- SS_PASTE@
- ELF_GET_CLIPBOARD@
- GR_PASTE@
- WP_PASTE@

ELF_SET_SELECTION@

Places a string array in the X selection buffer

Format ELF_SET_SELECTION@(stringArray)

Arguments stringArray A string array of text to be copied to the X Window selection buffer.

Description Places an array of text strings in the X Window selection buffer. This buffer can be accessed using paste actions.

ELF_STACK@

Returns the ELF execution stack

Format ELF_STACK@()

Description Returns an array containing the ELF execution stack. The current line of execution is the last element in the array. If you have a series of interconnected macros, there may be more than one element in the array. The following example shows a macro that uses ELF_STACK@, and the resulting array of information.

```
macro test
        test2()
endmacro


macro test2
        dump_array@(elf_stack@())
endmacro
```

```
line 10 in function TEST in file /newuser/test/macros/macro6.am
line 3 in function TEST2 in file /newuser/test/macros/macro6.am
```

Since TEST() calls, TEST2(), there are two lines in the ELF execution stack. For more information about the ELF Execution stack, see Chapter 5, "ELF Debugging", in the *ELF User`s Guide*.

ELF_STDOUT_RECORD@

Displays ELF keystroke recording in a shell window

Format ELF_STDOUT_RECORD@()

Description Prints the ELF code resulting from keystroke recordings directly to the window from which you started Applix*ware*. In this way, you can read the ELF code without having to actually save a macro using the Record Macro dialog box. You may want to include ELF_STDOUT_RECORD@ in your login.am file so that it is always in effect.

ELF_TASK_ID@

Returns the user's task ID

Format ID = ELF_TASK_ID@()

Description Returns the task ID of the currently executing ELF macro.

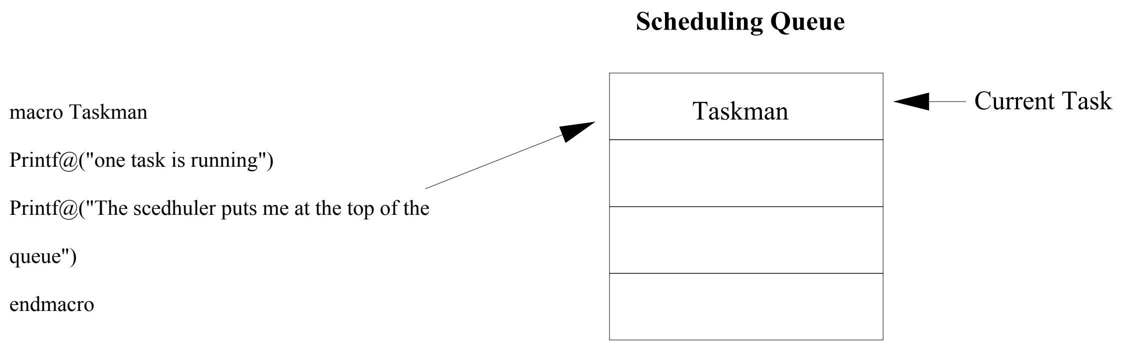
See also ELF_PARENT_TASK@

ELF Scheduler

The ELF scheduler maintains a queue of ELF tasks, and allocates time slices to each of the tasks in the queue.

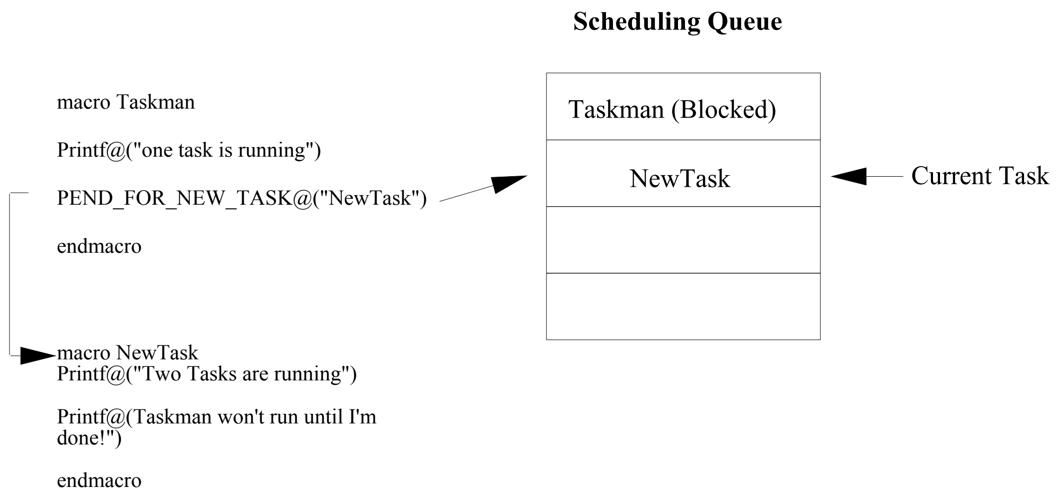
You can run more than one ELF task at a time. The macros NEW_TASK@, PEND_FOR_NEW_TASK@, and NEW_TASK_UNPENDED@ start new ELF tasks. When one of these macros runs, it adds a task to the *ELF scheduling queue*.

The ELF scheduling queue is a list of currently-running ELF tasks that is maintained by the ELF scheduler. When a task is added to the scheduling queue, it is added at the top of the queue. The following figure illustrates this.


Scheduling Queue
macro Taskman
Printf@("one task is running")
Printf@("The scedhuler puts me at the top of the queue")
endmacro
Taskman
Current Task

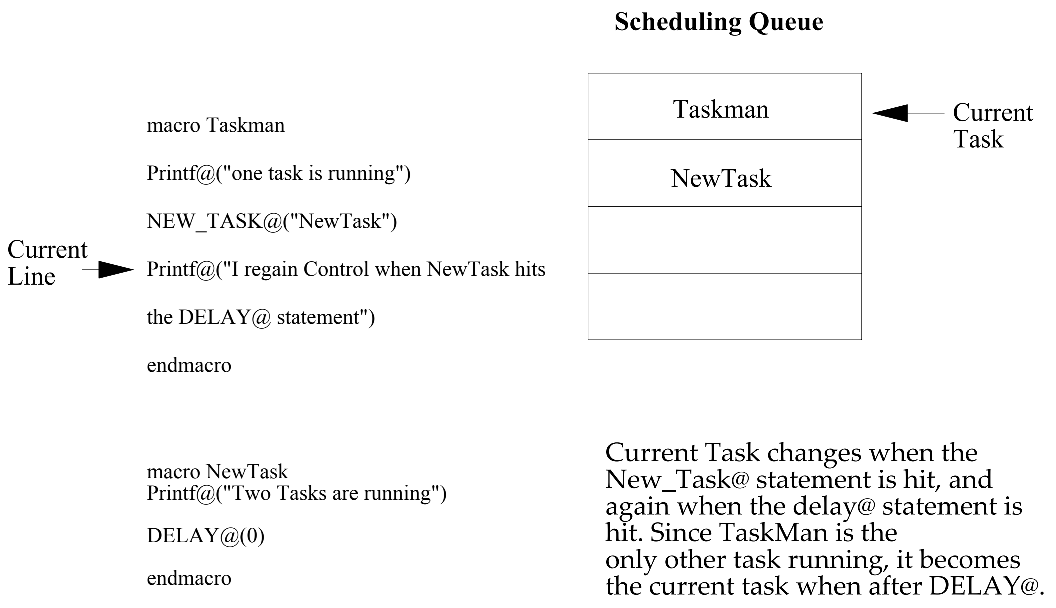
PEND_FOR_NEW_TASK@

If a running task spawns a new task, the new task is added to the queue. In the following diagram, the PEND_FOR_NEW_TASK@ macro suspends the execution of Taskman until its child task is completed.


Scheduling Queue
macro Taskman
Printf@("one task is running")
PEND_FOR_NEW_TASK@("NewTask")
endmacro
macro NewTask
Printf@("Two Tasks are running")
Printf@(Taskman won't run until I'm done!")
endmacro
Taskman (Blocked)
NewTask
Current Task

NewTask@

If you spawn a new task with the New_Task@ macro, the child task can give up control of the execution thread by calling either DELAY@, or DB_DISPLAY@. The following diagram shows this transaction.



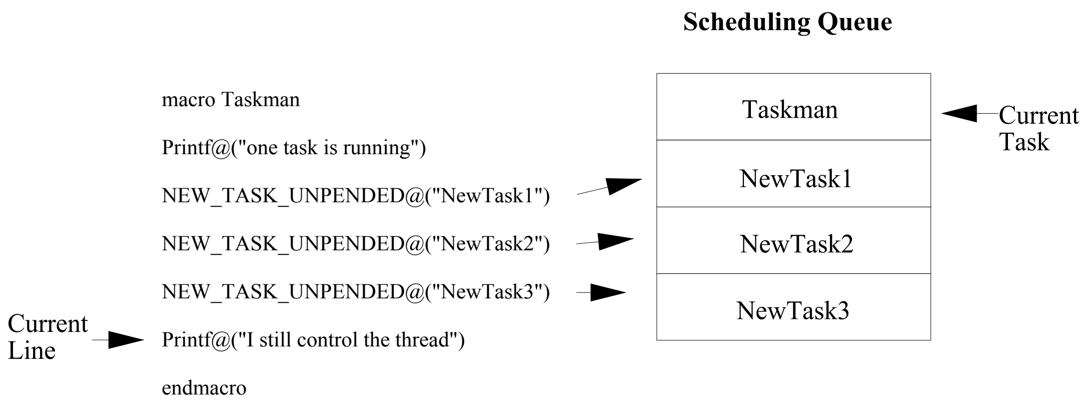
```
macro Taskman
Printf@("one task is running")
NEW_TASK@("NewTask")
Printf@("I regain Control when NewTask hits
the DELAY@ statement")
endmacro
```

```
macro NewTask
Printf@("Two Tasks are running")
DELAY@(0)
endmacro
```

Current Task changes when the New_Task@ statement is hit, and again when the delay@ statement is hit. Since TaskMan is the only other task running, it becomes the current task when after DELAY@.

New_Task_Unpended@

Both NEW_TASK@ and PEND_FOR_NEW_TASK@ give up control of the thread when they are called. You can add tasks to the scheduling queue, but maintain control of the thread with the macro NEW_TASK_UNPENDED@. The figure below shows a macro that adds three new tasks to the scheduling queue, but maintains control of the thread.



```
macro Taskman
Printf@("one task is running")
NEW_TASK_UNPENDED@("NewTask1")
NEW_TASK_UNPENDED@("NewTask2")
NEW_TASK_UNPENDED@("NewTask3")
Printf@("I still control the thread")
endmacro
```

The diagram shows that three thasks have been added to the scheduling queue, but the current task has not changed. When the Taskman macro exits, runs DELAY@ or runs DB_DISPLAY@, the scheduler changes the current task. Note that macros added to the scheduling queue using NEW_TASK_UNPENDED@ are run by the scheduler in the order that they are queued.

See also TASKS, NEW_TASK@, PEND_FOR_NEW_TASK@, NEW_TASK_UNPENDED@, DELAY@, DB_DISPLAY@

EOMONTH@

Returns the last day of the month
that falls a given number of months before or after a start date

Format EOMONTH@(startDate, months)

Arguments

startDate	A formatted date string.
months	a number of months. This value can be negative. Negative numbers are subtracted from the start date.

Description EOMONTH@ returns the date of the last day of the month that is a given number of months before or after the start_date.

The startDate argument is a number representing the date. This is the number of days the given date falls after 12/31/1899. This number is returned by the DATEVALUE@ macro.

If months is a positive integer, the EOMONTH@ function returns a date after the start date. if months is negative, the function returns a date before the start date.

EOMONTH@(DATEVALUE@("01/01/95"),1) equals 34758 02/28/95

EOMONTH@(DATEVALUE@("01/01/95"),-1) equals 34699 12/31/94

See also DATEVALUE@, EDATE@

EQN_SET_DEFAULT_POINTER@

Sets the new default pointer shape

Format EQN_SET_DEFAULT_POINTER@(pointerName)

Arguments pointerName The name of the new default pointer.

Description Selects the mouse pointer used by default. This is the pointer that is used within the Equation Editor's Window excluding its margin. After using this macro, the default pointer is changed for all equation windows.

If you wish to always use this pointer, it can be made permanent by setting a preference.

To obtain a list of available shape names, use **GET_CURSOR_LIST@**.

See also **EQN_SET_TEXT_POINTER@**

EQN_SET_TEXT_POINTER@

Sets the new text pointer shape

Format EQN_SET_TEXT_POINTER@(pointerName)

Arguments pointerName The name of the new text pointer.

Description Sets the new text pointer shape. This is the pointer that is used within the equation's margins. After using this macro, the text pointer is changed for all equation windows.

If you wish to always use this pointer, it can be made permanent by setting a preference.

To obtain a list of available shape names, use **GET_CURSOR_LIST@**.

See also **EQN_SET_DEFAULT_POINTER@**

ENTER_HELP_MODE@

Places an application in help mode

Format ENTER_HELP_MODE@()

Description Places an application in *help on context* mode and changes the cursor shape to the help cursor. While in *help on context* mode, the next menu selection will display the help screen for that selection and end help mode.

ERROR@

Throws an error

Format ERROR@(code, string[,object])

Arguments		
	code	A unique error number. code must be in the range of 1 to 1000.
	string	The error string that is displayed in an error dialog box.
	object	A string representing the object of an error.

Description ERROR@ throws the error you specify.

See also ERROR_BOX@
ERROR_FILE@
ERROR_FUNCTION@
ERROR_LINE@
ERROR_NUMBER@
ERROR_OBJECT@
ERROR_RETHROW@
ERROR_STRING@

ERROR_BOX@

Displays an error dialog box

Format ERROR_BOX@(code, string[, object])

Arguments		
	code	A unique error number. code must be in the range of 1 to 1000.
	string	The error string that is displayed in an error dialog box.
	object	A string representing the object of an error.

Description Displays an error dialog box that shows the error string and error object for the most recently thrown error. Use ERROR_BOX@ within an error handler to display the most recently thrown error.

Example

See also ERROR@
ERROR_FILE@
ERROR_FUNCTION@
ERROR_LINE@
ERROR_NUMBER@
ERROR_OBJECT@
ERROR_RETHROW@
ERROR_STRING@

ERROR_FILE@

Returns the source file of the most recently thrown error

Format filename = ERROR_FILE@()

See also ERROR@
ERROR_BOX@
ERROR_FUNCTION@
ERROR_LINE@
ERROR_NUMBER@
ERROR_OBJECT@
ERROR_RETHROW@
ERROR_STRING@

ERROR_FUNCTION@

Returns the last error macro name

Format macroName = ERROR_FUNCTION@()

Description Returns the name of the macro that caused the last error throw to occur.

ERROR_LINE@

Returns the line number of the most recently thrown error

Format lineNum = ERROR_LINE@()

See also ERROR@
ERROR_BOX@
ERROR_FILE@
ERROR_FUNCTION@
ERROR_NUMBER@
ERROR_OBJECT@
ERROR_RETHROW@
ERROR_STRING@

ERROR_NUMBER@

Returns the error code of the most recently thrown error

Format num = ERROR_NUMBER@()

Example

See also ERROR@
ERROR_BOX@
ERROR_FILE@
ERROR_FUNCTION@
ERROR_LINE@
ERROR_OBJECT@
ERROR_RETHROW@
ERROR_STRING@

ERROR_OBJECT@

Returns the error object for the most recently thrown error

Format object = ERROR_OBJECT@()

See also ERROR@
ERROR_BOX@
ERROR_FILE@
ERROR_FUNCTION@
ERROR_LINE@
ERROR_NUMBER@
ERROR_RETHROW@
ERROR_STRING@

ERROR_RETHROW@

Rethrows the most recently thrown error

Format ERROR_RETHROW@()

Description Rethrows the number, string, and object of the most recently thrown error. It is useful when you want to handle certain errors but ignore (rethrow) others. If no errors have been thrown, ERROR_RETHROW@ throws the error ERR#ILLVAL_.

Example

See also ERROR@
ERROR_BOX@
ERROR_FILE@
ERROR_FUNCTION@
ERROR_LINE@
ERROR_NUMBER@
ERROR_OBJECT@
ERROR_STRING@

ERROR_STRING@

Returns a string value of an error

Format ERROR_STRING@([errorNumber])

Arguments

errorNumber	A number value indicating the error to which the error string belongs. If you do not include errorNumber, then the error string associated with the most recently thrown error is returned.

Description Returns a string value representing the error message associated with a specified error number. errorNumber can be an ELF error number or an error number you have defined.

See also ERROR@

ERROR_BOX@

ERROR_FILE@

ERROR_FUNCTION@

ERROR_LINE@

ERROR_NUMBER@

ERROR_OBJECT@

ERROR_RETHROW@

EXIT@

Cancels the execution of the current macro

Format EXIT@([data])

Arguments

data	Any ELF data including formats and arrays.

Description Cancels the execution of the current macro and all ELF macros in the executing task. It does not close any operating system files opened by the macro from which it is called.

If you have invoked the current task using PEND_FOR_NEW_TASK@, you can return data to the invoking task using the optional data argument. In this case, EXIT@ is functioning similar to the RETURN statement.

See also EXIT_ALL@

EXIT_FUNC@

EXIT_ALL@

Exits all Applix*ware* windows

Format EXIT_ALL@()

Description Exits all opened Applix*ware* applications and then closes the main menu. EXIT_ALL@ is similar to EXIT@, except you are given the option of saving your work before the exit. This macro does not close any operating system files opened by a macro.

Example

See also EXIT@
EXIT_FUNC@
LOGOUT@

EXIT_FUNC@

Cancels the execution of the current function

Format EXIT_FUNC@()

See also EXIT@
EXIT_ALL@

EXP@

Raises 'e' to the power indicated

Format EXP@(power)

Arguments power A number indicating the desired power.

Description Raises the number *e* (approximately 2.718282) to the power indicated by the value entered as an argument. The formula EXP@(6) raises *e* to the 6th power. It returns the value 403.42879.

See also POWER@

FILE_EXECUTABLE@

Checks whether the user has permission to execute a file

Format FILE_EXECUTABLE@(filename)

Arguments

filename	The name of the file to check for permission to execute.

Description Throws the error ERR#ERR_FEEACCES_ if the user does not have execution permission for file name. If a user has write permission, macro execution continues with the statement after FILE_EXECUTABLE@.

See also FILE_READABLE@

FILE_WRITABLE@

FILE_EXISTS@

Checks to see if a file exists

Format flag =FILE_EXISTS@(filename)

Arguments

filename	The name of the file to check if it exists.

Description Checks to see if a file exists and returns a Boolean value. TRUE means that the file exists.

Example

FILE_HEADER_FROM_INFO@

Creates the file header line information

Format headerLine = FILE_HEADER_FROM_INFO@(format afile_info info)

Arguments

info	A data structure containing information about a file. The definition of this format is as follows:

```
format afile_info
    encoding, 'TRUE if file is encoded
    version,   'version number of current format
```

```
docType,  'as in regcfil_.am
          'version number of original document
original_version,
          'last version capable of reading this file
minimum_version,
          'arbitrary hint string
content_hint
```

Description Creates the file header information that is placed at the beginning of Applix*ware* files. For example, it could create and return the following string:

*BEGIN WORDS VERSION=400/320 ENCODING=7BIT

FILE_PERMISSIONS@

Returns an array of file permissions

Format permArray = FILE_PERMISSIONS@([fileName])

Arguments fileName The full path name and file extension of a system file.

Description Returns an array of file permissions for fileName. If fileName is not supplied, does not exist, or is NULL, the umask permissions are returned. permArray[0] is the *group* file permissions and permArray[1] is the *all* file permissions. The file permissions are one of the following values:

0	No read or write permissions for the file.
1	Read permission for the file
2	Read and write permissions for the file.

FILE_READABLE@

Checks whether a user has permission to read a file

Format FILE_READABLE@(filename)

Arguments filename The name of the file to check for permission to execute.

Description Throws the error ERR#ERR_FEEACCESS_ if the user does not have read permission for file name. If a user has read permission, macro execution continues with the statement after FILE_READABLE@.

To check for the existence of a file, see **LIST_OF_FILES@**.

See also FILE_EXECUTABLE@
FILE_WRITABLE@

FILE_SAVED_BROADCAST@

Notifies tasks that a file is saved

Format FILE_SAVED_BROADCAST@(filename)

Arguments

filename	The name of the file that was saved.

Description Sends a message to all tasks that indicates that filename is saved.

FILE_SIZE@

Returns the size of a file

Format size =FILE_SIZE@(filename)

Arguments

filename	The name of the file whose size you are checking.

Description Returns the size of a file. If the file does not exist, this macro returns -1.

Example

FILE_SYSTEM_DIRS@

Lists all directories

Format dirArray = FILE_SYSTEM_DIRS@(dir, wildcard)

Arguments

dir	The directory in which you are searching for other directories.
wildCard	Specifies the file name character matching criteria. Wildcard characters may be used:

*	Match all characters
?	Match any single character
[ ]	Match any single character in brackets

[?] Match any single character except those in brackets

Description Lists all directories matching a wildcard specification in directory dir. Files are not listed.

See also FILE_SYSTEM_FILES@
LIST_OF_DIRS@

FILE_SYSTEM_FILES@

Lists all files

Format fileArray = FILE_SYSTEM_FILES@(dir, wildcard[, doDotsFlag])

Arguments

dir	The directory in which you are searching for other files.
wildCard	Specifies the file name character matching criteria. Wildcard characters may be used:
	* Match all characters
	? Match any single character
	[ ] Match any single character in brackets
	[?] Match any single character except those in brackets
doDotsFlag	A Boolean value which if set to TRUE indicates that dot files are included in the listing.

Description Lists all files matching a wildcard specification in a directory. Directories are not listed.

NOTE: This macro generates considerable file system overhead.

See also FILE_SYSTEM_DIRS@
LIST_OF_FILES@

FILE_SYSTEM_INFO@

Returns information about files and directories

Format infoArray = FILE_SYSTEM_INFO@(dir, wildCard, recursiveFlag, nameOnlyFlag, sortMode, changeDirFlag[, doDotsFlag])

Arguments

dir	The directory from which information should be gathered.
wildCard	Specifies the file name character matching criteria. Wildcard characters may be used:

*	Match all characters
?	Match any single character
[ ]	Match any single character in brackets
[?]	Match any single character except those in brackets

recursiveFlag

When TRUE, sub-directories of a directory are searched recursively for files matching search_criteria. If FALSE, sub-directories are not searched.

nameOnlyFlag

When TRUE, only the file name and file type of files matching searchCriteria and the directory names, are returned. Each name and its file type are returned as a two-dimensional array element in which the first element in the two-dimensional array is the file name and the second is a number indicating the file type. If FALSE, all file and directory description information is returned for each file matching searchCriteria.

All directory names in the specified directory are returned regardless of search criteria. The file names returned include the file name extension only if they are non-Applix*ware* documents.

sortMode The order in which the file description information is returned. Possible values for sort_mode are:

0	unsorted
2	Sorted by date, oldest first
3	Sorted by date, most recent first
4	Sorted alphabetically by name, a to z
5	Sorted alphabetically by name, z to a

changeDirFlag

Indicates whether dir should be made the current directory. If TRUE, dir is made the current directory. If FALSE, dir is not made the current directory.

doDotsFlag A Boolean value which if set to TRUE indicates that dot files are included in the listing.

Description If nameOnly is TRUE, FILE_SYSTEM_INFO@ returns a string array of the names of the files or directories matching the search_criteria and their file types. Otherwise, FILE_SYSTEM_INFO@ returns an array containing the following information for each file or directory matching the search criteria:

name The relative name of the file or directory.

type One of the following types of files or directories:

DEFINITION	**NUMBER**	**DESCRIPTION**
FSMAT#DIRECTORY	-1	System directory
FSMAT#FILE	0	System file
FSMAT#TEXT_	1	Word Processor

FSMAT#PICTURE_	2	Graphics Editor, Presents
FSMAT#SPREADSHEET_	3	Spreadsheet
FSMAT#COMMAND_	4	ELF
FSMAT#DIALOG_	8	Dialog box
FSMAT#BITMAP_	9	Applixware Bitmap
FSMAT#QUERY_	10	Query
FSMAT#EQUATION_	12	Equation
FSMAT#BUILDER_	13	Applixware Builder
FSMAT#BUILDER_TURBO_	15	Applixware Builder Turbo
FSMAT#SLIDE_SHOW_	16	Applixware Slide show

last_modified
: The date/time that the file or directory was last modified. This date/time is based on the number of seconds elapsed since January 1, 1970 GMT. Use **DATE_FORMAT@** to change this number into a readable date/time.

kbytes
: The size, to the closest kilobyte, of the file.

remainder_bytes
: Adding remainder_bytes to the kbytes amount will give you the exact size of the file.

mode A 9-character string representing the access mode for the file or directory. Characters 1-3 are the read, write, and execute modes for the owner, characters 4-6 are the read, write, and execute modes for the group, and characters 7-9 are the read, write, and execute modes for others.

links The number of hard UNIX links.

owner
: The name of the owner of the file.

All arguments to this macro are optional. If you omit dir, the default is the current directory.

This macro can use considerable file system resources to generate its information.

NOTE: If more than 1,000 files exist within a directory, FILE_SYSTEM_INFO@ will fail. You can correct this problem by setting the three Directory Display Preference options:

Maximum Number of Files to Search through - Using this option, you can raise the limit to 20,000.

Maximum Levels in a Recursive Search - This sets the depth to be 32,000.

Maximum Search Time in Seconds - Tracks the length of time that a search can take

.

Example

See also LIST_OF_DIRS@
LIST_OF_FILES@

FILE_WRITABLE@

Checks whether a user has permission to write to a file

Format FILE_WRITABLE@(filename)

Arguments

filename	The name of the file to check for write permission.

Description Throws the error ERR#ERR_FEEACCES_ if the user does not have write permission for file name. If a user has write permission, this macro continues execution with the next statement. To check for the existence of a file, see LIST_OF_FILES@.

See also FILE_EXECUTABLE@
FILE_READABLE@
SAVE_CHECK@

FILTER_DIR@

Returns the name of the filter directory

Format dir = FILTER_DIR@()

Description Returns the name of the directory in which the filter programs reside.

FILTER_DOC_TO_CURRENT@

Converts file to a format that can be opened in current revision

Format infile = FILTER_DOC_TO_CURRENT@(infile)

Arguments

infile	The name of the file being converted.

Description Converts the format of infile so that it can be opened by the application. This application does not have to be the current release. It only has to be release 3.2 or later.

If the returned file name differs from the input file name, the calling application will need to "shift the name" of the file so that a warning message is posted if the user tries to save the converted file using the same name as the original file.

FIND_MACRO_FILE@

Searches standard directories for an ELF file

Format path name = FIND_MACRO_FILE@(filename)

Arguments

filename	The name of the ELF file to find.

Description Searches the /userHome/axhome/macros directory for the specified file. If the file exists, the macro returns the full path name of the file. If the file does not reside in the user's macros directory, FIND_MACRO_FILE@ returns NULL.

Note that this macro does not examine the directories in the ELF Search List.

See also FIND_USER_CUST_SYS_ELF_FILE@

FIND_REGISTERED_OBJECT@

Returns an object reference to a registered object

Format FIND_REGISTERED_OBJECT@(name, startFlag)

Arguments

name	The registered name of an Applixware object.
startFlag	If TRUE, Applix*ware* launches the Builder program containing the object. If FALSE, Applix*ware* does not launch the Builder object.

Description Returns a Builder object reference to the named object. The object must have been previously registered with the macro REGISTER_OBJECT@.

FIND_USER_CUST_SYS_ELF_FILE@

Searches directories for a file

Format path name = FIND_USER_CUST_SYS_ELF_FILE@(file)

Arguments file The name of the file to find.

Description Searches through a user's User, Custom, System, and Macros directories for a specified file. If file is found in one of these directories, file's full path name is returned. If file is not found and a Macros directory exists, FIND_USER_CUST_SYS_ELF_FILE@ returns the full path name as if file were in the Macros directory. If file is not found in any of the directories and no Macros directory exists, FIND_USER_CUST_SYS_ELF_FILE@ returns the full path name as if file were in the System directory.

See also FIND_USER_CUST_SYS_FILE@

FIND_USER_CUST_SYS_FILE@

Searches directories for a file

Format path name = FIND_USER_CUST_SYS_FILE@(file)

Arguments file The name of the file to find.

Description Searches through a user's User, Custom, and System directories for a specified file. If file is found in one of these directories, file's full path name is returned. If file is not found in any of the directories FIND_USER_CUST_SYS_FILE@ returns the full path name as if file were in the System directory.

See also FIND_USER_CUST_SYS_ELF_FILE@

FORMAT@

Formats strings

Format string = FORMAT@(format, arg1[,arg2, arg3,...argN])

Arguments format A string that includes format specifications.
1. ..argN One or more text strings or numbers to be inserted in the format string.

Description Returns a formatted string based on the format specifications and the arguments provided. The format string can contain text and one or more format specifications. For example:

FORMAT@("The answer is %s.",stringArg)

If the value of stringArg is "New York", the returned string is:

The answer is New York.

In this example, the letter 's' is a conversion indicator that specifies that the argument contains a string.

A format specification is as follows:

%[flag][width][.precision]specifier

The following paragraphs describe this format in more detail:

% Symbol

A % (percent) symbol indicating that the information that follows is a format.

You can also use the format %<number>$. Normally, format specifications correspond one-to-one with arguments. This format allows you to specify format specifiers in any order. That is, %3$s indicates that the third argument is to be used for this format. For example:

FORMAT@("The integer portion of %1$6.3d is %1$6d", arg1)

You can defer evaluating the width or precision components by specifying *<number>$.

If either of these two formats are used, all format specifiers must use them.

Flag Character

An optional flag character, as follows:

-	The number will be left justified
+	A plus or minus sign is always printed
(blank)	If the first character is not a plus or minus sign, a blank space is displayed in place of a + result.
#	Convert to an alternate form. (This is only used by 'e', 'E', 'f', 'g', 'G', 'o', "O', 'x', and 'X' formats.) For 'o' and 'O', a lead 0 (zero) is displayed. For 'x' and 'X', the value is preceded by either '0x' or '0X'. For the other formats, the output will always contain a decimal point even when there are no characters following the decimal point. For 'g' and 'G' conversions, trailing zeroes are not removed.

Width

An optional number indicating the minimum number of characters to be used when displaying the argument. Arguments are never truncated; however, if the argument is smaller than this width, the argument will be padded with characters to the right or the left, depending on how the argument is justified within this width.

Instead of specifying a number, you can type an asterisk (*). When the format is interpreted, the width is taken from the corresponding argument. For example:

FORMAT@("Value 1 is %4d; Value 2 is %#d", arg1, arg2, arg3)

Here, the width of the second argument is indicated by arg2. If this number is negative, numbers are left justified.

Precision

A precision that indicates the minimum number of digits that will appear in a numeric conversion. (Precision indicates the number of digits that appear after a decimal point.) For example %4.2d indicates that a minimum of four character positions is reserved. Of these four, two will be used to represent the fractional component. That is, 99.738 is displayed as 99.74. 99.7 is displayed as 99.70. However, 199.738 is displayed as 199.74 (which used 5 digits.)

Instead of specifying a number, you can type an asterisk (*). If the converted number is negative, the value is changed to 0.

Conversion Indicators

Here is a list of all supported conversion indicators:

Character	Converts:
c	number (or character) to a character
d	integer to signed decimal.
e	floating point or double is converted to a string in the following style: d.dddde±ddd. The number of digits. The precision used to display the number (which is described below) indicates the number of digits after the decimal point.
E	floating point or double is converted to a string in the following style: d.ddddE±ddd. The number of digits. The precision used to display the number (which is described below) indicates the number of digits after the decimal point.
f	floating point or double is converted to decimal. The default precision is six digits.
g	floating point or double is printed in either 'f' or 'e' style. 'e' style is used if the exponent will contain more than three digits
G	floating point or double is printed in either 'f' or 'E' style. 'E' style is used if the exponent will contain more than three digits
i	integer to signed decimal.
o	integer to unsigned octal.
p	integer to unsigned hexadecimal; hexadecimal letters (a-f) are in lowercase.
s	string to string
u	integer to unsigned decimal.
x	integer to unsigned hexadecimal; hexadecimal letters (a-f) are in lowercase.
X	integer to unsigned hexadecimal; hexadecimal letters (a-f) are in uppercase.
%	Prints a %.

If the argument does not contain a numeric value, Applix*ware* will throw an error.

Any argument can be an expression. For example:

FORMAT@("My value is %9d", A/B)

where both A and B are variables. If the value were undefined (because the value of B was zero), the returned value would be the string Infinity. In other cases where the value is undefined, the value NaN (Not a Number) is returned.

General Information

If a field width number is not included in a format specification, the width of the field is the same as the width of the arg to which the format specification applies.

No argument is ever truncated if you do not indicate enough characters for it. For example, if you indicate that a string should display using 10 characters and the string is 20 characters long, all 20 characters are displayed.

A field width can be specified using an asterisk (*) instead of a number. If you use an asterisk, an integer argument indicates the length of the field. For example:

```
format(``%*s'', width, ``foo'')
```

When FORMAT@ encounters the first format specification, it substitutes arg1 in place of the specification and formats arg1 according to the specification. If a second format specification is encountered, arg2, is substituted for the format specification. Subsequent format specifications apply to subsequent args in the args list. If there are more format specifications in a format string than there are args in the args list, the empty string is substituted for any format specifications not having args.

Example

FORMAT_ARRAY@

Returns an array whose elements are not longer than a specified length

Format newArray = FORMAT_ARRAY@(array, elementLength)

Arguments

array	The ELF array being reformatted. The array may contain strings or numbers.
elementLength	The maximum length (in characters) of each element in the new array. Characters in excess of this length form new element(s).

Description Reformats array so that each element does not exceed a specified length in characters. If you want to read an ASCII file in a shell window whose displayed line width is shorter than the longest line of ASCII text, you could reformat the array of text lines to conform to the current shell window. FORMAT_ARRAY@ can also be used to wrap list box items whose length exceeds the list box's displayed line width.

See also ARRAY_APPEND@
ARRAY_TRANSPOSE@
SORT@

FREE_NAMED_BITMAP@

Frees up named bitmaps from the Pixmap cache

Format FREE_NAMED_BITMAP@(name)

Arguments		
	name	The bitmap that you want to release from the Pixmap cache.

Description Releases a bitmap from the Pixmap cache.

FULL_PATHNAME@

Converts a relative path name to an absolute path name

Format fullPathname = FULL_PATHNAME@(relPathname)

Arguments		
	relPathname	The name of the path name you want to convert.

Description Converts a relative path name to an absolute path name.

GET_CURSOR_LIST@

Returns an array of cursor symbol names

Format GET_CURSOR_LIST@(nameArray)

Arguments		
	nameArray	The variable to which the string array is returned.

Description Creates a string array containing the names of every cursor symbol available in Applix*ware*.

Example

GET_DLG_FONT_HEIGHT@

Returns the height of a font at the current point size

Format fontHeight = GET_DLG_FONT_HEIGHT@()

GET_DISPLAY_HEIGHT@

Determines a screen's height

Format height = GET_DISPLAY_HEIGHT@()

Description Returns an integer that shows the height of the screen device.

Example

See also GET_DISPLAY_WIDTH@

GET_DISPLAY_WIDTH@

Determines a screen's width

Format width = GET_DISPLAY_WIDTH@()

Description Returns an integer that shows the width of the screen device.

Example

See also GET_DISPLAY_HEIGHT@

GET_ENV_VAR@

Returns an operating system environment variable`s value

Format value = GET_ENV_VAR@(envVar)

Arguments envVar A string containing the name of an operating system environment variable.

Description Returns the value of an operating system environment variable. If the variable is not defined or does not have a value, this macro returns NULL.

Example

GET_GRAPHIC@

Returns a task's graphic handle

Format GET_GRAPHIC@(gfx, taskID)

Arguments

gfx	The handle for the graphic editor. This is a returned value.
taskID	A task ID.

Description Returns the graphics handle gfx associated with a task.

GET_HELP_TEXT@

Returns an array of strings from the help database

Format array = GET_HELP_TEXT@(name)

Arguments

name	The help ID in the help database.

Description This function returns an array of strings from the help database. This is the information you have placed into the help system. Information for the on-line hypertext system is not included.

NOTE: This macro will become obsolete in a future release.

GET_LANGUAGE@

Returns the current language

Format GET_LANGUAGE@()

Description Returns the current setting of the AXLANG environment variable. The AXLANG environment variable is used to establish the language in which the Applixware menu bars and help appears. The following table lists the values for AXLANG, and the corresponding language:

Value of AXLANG	**Language**
1	English
2	German
3	French

GET_LINKS_INFO@

Returns an array of link information

Format format doc_links_info@ info = GET_LINKS_INFO@(filename)

Arguments filename The name of the file from which you are obtaining link information.

Description Returns a doc_links_info@ format that describes the documents linked to filename. The doc_links_info format contains the following fields:

```
format doc_links_info@
        format arrayof link_info@ links,
        format arrayof object_link_info@ objects


format link_info@
        name,                           ' string:     linked file name
        docType,                        ' int: doc type returned by
                                        ' RECOGNIZE_FILE_INFO@
        appType,                        ' int: app type returned by
                                        ' RECOGNIZE_FILE_INFO@
        filterMacro,                    ' string:     import filter macro
        launchMacro,                    ' string:     macro to launch application
        externalRefName,        ' string:     reference in linked doc
        internalRefName,        ' string:     reference in parent doc
        format arrayof doc_links_info@ info,
        override                ' string:     location of nested doc w/links

format object_link_info@
        name,                   ' string:     object name
        docType,                ' int: doc type returned by
                                ' RECOGNIZE_FILE_INFO@
        appType,                ' int: app type returned by
                                ' RECOGNIZE_FILE_INFO@
        filterMacro,            ' string:     import filter macro
        launchMacro,                    ' string:     macro to launch application
        internalRefName,        ' string:     reference in parent doc
```

```
allowUnreferenced,          ' int: TRUE->save even if not referenced
format arrayof doc_links_info@ info
```

See also GET_LINKS_LIST@
SS_GET_LINKS@
WP_GET_LINKS_INFO@

GET_LINKS_LIST@

Returns a list of link information

Format linkArray = GET_LINKS_LIST@(filename)

Arguments filename The name of the file from which you are obtaining link information.

Description Returns an array that contains a list of all documents linked to this document. The format of the returned data is **link_info@**. Unlike **GET_LINKS_INFO@**, the information returned by this macro is a flat list.

GET_LINKS_LIST_FROM_INFO@

Flattens a doc_links_info@ format

Format linksList = GET_LINKS_LIST_FROM_INFO@(filename, format doc_links_info@ info)

Arguments

filename	The name of the file from which you are obtaining link information.
info	An array of **docs_link_info@** information. See **GET_LINKS_INFO@** for more information.

Description Transforms doc_links_info@ formatted information into **link_info@** formatted information. Both sets of information contain all information about a document's links. However, the doc_links_info@ format embeds other link information within itself if a link contains a link.

The link_info@ format *unwinds* this information so that the information is a simple linear array rather than a recursive data structure. Working with doc_links_info@ information is more difficult than with link_info@ information because of the recursion. However, because doc_links_info@ information must be transformed into link_info@ format (and vice versa), working with doc_links_info@ information is faster than with link_info@ information.

GET_MAX_FILE_LENGTH@

Returns the maximum length of a file's name

Format length = GET_MAX_FILE_LENGTH@(dir)

Arguments	dir	An arbitrary directory name.

Description Returns the largest length that can be used when creating a file name. This macro's results may be inaccurate if the directory name is on a file system mounted using NFS.

GET_MESSAGE@

Returns message information

Format array = GET_MESSAGE@(timeout)

Arguments	timeout	The time in seconds. If you set this value to -1, no timeout occurs.

Description Wait for message from another task. If your ELF task is acting as a client, this message is probably in response to sending a request off to a server task. If your task is acting as a server, this message is probably a request to perform a service for another task.

This macro returns an array of 2 elements:

· The first element is the command code.

· The second is any required data.

GET_PAGE_INFO@

Returns information about pages to be printed

Format numArray = GET_PAGE_INFO@(string)

Arguments	string	The name of the string entered by the user.

Description Takes string and decomposes it into page numbers and their extensions. GET_PAGE_INFO@ then returns an ELF array of five integers, as follows:

· Status

· Starting page

· Ending page

- Starting page extension
- Ending page extension

If status is not 0, then an error occurred and the rest of the elements in the array have no meaning. The following are examples of valid strings:

1a - 1c
3a - 1z
999a- 9999zz
1- 2

GET_RESOURCE@

Locks a resource

Format GET_RESOURCE@(name)

Arguments	name	The name of a resource. This name is defined (and agreed to) by macros that use the resource.

Description Locks resource name. If the resource is locked when this macro executes, it waits for the current holder of the resource to release it. (That is, the second task requiring the resource is blocked.) There is no timeout provision, which could mean a long wait until the resource is released. (A task releases a resource by calling **RELEASE_RESOURCE@**.)

GET_TEMP_FILENAME@

Creates a temporary file name

Format filename = GET_TEMP_FILENAME@()

Description Creates a temporary file name if one is not already created. A similar and more useful macro is **CREATE_TEMP_FILE@**.

GET_TEMPLATE_DIR@

Returns the directory containing
the Applixware Presents templates

Format GET_TEMPLATE_DIR@()

Description Returns the directory containing the templates used by Applixware Presents. By default, this directory is /*install_dir*/template.

GETDISPLAYVAR@

Returns the current X display

Format XdisplayVar = GETDISPLAYVAR@( )

Description Returns the value of the current X display. In the C Shell, this is the value assigned to the DISPLAY environment variable.

GETHOSTNAME@

Returns the name of the host

Format hostname = GETHOSTNAME@( )

HELP_CHECK_LINKS@

Checks that hypertargets have hyperlinks

Format HELP_CHECK_LINKS@(topicFile)

Arguments		
	topicFile	The full path name of the file named topics. This file was created by the BuildHyperFiles macro within BldHelp.am.

Description Examines all of the files within the file named filelist. (This file must be in the same directory as topicFile.) As it scans these files, it creates a list of all hyperlinks. (See **HELP_HYPERLINK@**.) These hyperlinks are then compared against the target list within topicFile.

A list of problems is placed into an empty Words document.

See also **HELP_FIND_DUP_LINKS@**

HELP_COPY_TOPIC@

Copies the current Help topic to the clipboard

Format HELP_COPY_TOPIC@()

Description Copies the entire topic to the clipboard. If the text has fields, these fields are also copied. This could mean that the copied topic refers contains hypertext links and inset fields that will not work properly when pasted.

Example

HELP_DEFAULT_WINDOW_SIZE@

Resets the Help window's size back to its original size

Format HELP_DEFAULT_WINDOW_SIZE@()

Description Sets the Help window's length and width back to their default size. This macro is normally used after the Help window's size was changed using HELP_SET_WINDOW_SIZE@.

The default size is 6050 (width) x 7000 (height).

HELP_FIND_DUP_LINKS@

Finds duplicate hypertarget names

Format HELP_FIND_DUP_LINKS@(topicFile)

Arguments

topicFile	The full path name of the file named topics. This file was created by the BuildHyperFiles macro within BldHelp.am.

Description Examines topicFile and displays a list of duplicate hypertarget names. This check is not perfect as it will not tell you if one of the hypertargets in this domain has the same name as a hypertarget in any other domain including the ones supplied with Applix*ware*.

See also HELP_CHECK_LINKS@

HELP_FREEZE_DOMAIN@

Sets the Help domain and keeps it there

Format HELP_FREEZE_DOMAIN@(domain)

Arguments

domain	A previously registered help domain.

Description Sets the Help domain and prevents the user from changing to another domain while using the Search dialog box. If the Search dialog box is displayed when this macro executes, domain will become the current domain.

While other entries on the domain pull down within the Search dialog box are still displayed, the user cannot switch to them.

See also HELP_UNFREEZE_DOMAIN@

HELP_GET_HELPSTYLE_TEXT@

Returns a Help topic's topic line

Format rawText = HELP_GET_HELPSTYLE_TEXT@(format wp_range@ glossaryRange, beadNum)

Arguments

glossaryRange	The range of the Help glossary within a file.
beadNum	The bead number of a bead within a Help topic.

Description Returns the text of the helpStyle@ paragraph of a topic. The topic is identified by passing the bead number of any bead within the topic.

This macro is visible from the Help build procedure. It is not normally used.

HELP_GET_INDEX_ENTRIES@

Returns index information in a file

Format format HELPrawIndex@ rawIndex = HELP_GET_INDEX_ENTRIES@(filename)

Arguments

filename	A Words file containing index entries.

Description Returns a format containing all index information contained within a file. This macro is executed within the BuildHyperFiles macro (contained within BldHelp.am) and is seldom executed outside of this context.

The format of HELPrawIndex@ is as follows:

```
format HELPrawIndex@,
        filename,
        domain,
        format arrayof HELPlink@ term
```

The format of HELPlink@ is:

```
format HELPlink@
        topic,
        index
```

HELP_HYPERLINK@

Hyperlink macro for user-defined Help

Format HELP_HYPERLINK@(argArray)

Arguments argArray The following two-element array:

arg[0]	the hypertarget
arg[1]	the domain of the hypertarget

Description Calls the Applixware Search engine to locate a hypertarget and display it within a help window. This macro is usually placed within a Words document as the following macro field:

```
{macro "help_hyperlink@" -pass"<link>" -pass "<domain"
                -text "<Visible Link Text>"
                -textStyle "<menu>"}
```

However, it may be invoked within other macros when you are placing the invoking command on menu bars.

Example

See also HELP_CHECK_LINKS@

HELP_FIND_DUP_LINKS@

HELP_POP_UP@

Displays a Pop-up Help window

Format	HELP_POP_UP@(gloss_text)	
Arguments	gloss_text	The topic name of the text to be displayed.
Description	Displays the text associated with a pop-up name in a scrolling text box.	

HELP_POSITION_AT_INDEX@

Cursor is put at the index term rather than at beginning of topic

Format HELP_POSITION_AT_INDEX@(domain)

Arguments domain One of the domains registered with Help using **HELP_REGISTER@**.

Description Tells Help that it should place the cursor at the position of the search (index) term rather than at the beginning of the topic. When this occurs, the default behavior is to highlight the line containing the index field.

If the preference helpHighlight is set to the domain name, then the macro whose name is assigned to the helpHighlightDomainHook is executed after the cursor is placed at the index field.

If a highlighting macro is set, Help does not highlight the line.

HELP_REGISTER@

Adds a domain to Help

Format HELP_REGISTER@(domain, rootPath[, searchFile[, contentsFile, testFlag]]])

Arguments	domain	The name of the domain. This name is the same as the name assigned to the HELPdomain@ document variable within each document.
	rootPath	The place in the file system containing all of your .ah files, your topics file, and <domain_name> file.
	searchFile	The name of the file containing the search entries. This file is created when you build Help.

	contentsFile	The name of the file containing a hyper_target whose value is Contents. When the user presses the Contents button within the Help window, Help will access this topic.
	testFlag	An optional flag that displays diagnostic information when HELP_REGISTER@ is run. The domain will not be registered if this flag is set.

Description Informs Help that another domain has been added to it and also specifies where it can find this domain's information. No Help information can be obtained from a domain until it is registered.

No domain can be registered more than once. Use **HELP_UNREGISTER@** if you need to remove a domain from Help. (This is seldom necessary.)

HELP_RESET@

Removes all domain knowledge

Format HELP_RESET@()

Description Sets all Help system variables and data structures to NULL. When you are creating and debugging help related macros, use this macro to reset Help back to initial state.

HELP_RESET_SIZE@

Redefines the Help window's size to be its default

Format HELP_RESET_SIZE@()

See also **HELP_DEFAULT_WINDOW_SIZE@**

HELP_SET_APP_NAME@

Sets the Help window's title for an application

Format HELP_SET_APP_NAME@(domain, windowTitle)

Arguments	domain	A domain registered with Help using **HELP_REGISTER@**.
	windowTitle	The name to be set at the top of the window.

Description	Assigns a name to be used for a window when Help is displayed for a domain. Other domains still will use the help window. That is, setting a new window title causes Help to be displayed in more than one window. However, all the information for a domain must appear in one window. For example, if this macro is set for the ELF domain, all ELF information would appear in an ELF Help window. The other Applixware applications would continue to use the same window. This macro is seldom used with the Applixware built-in domain. Instead, it can be used when you are creating your own Help applications and you wish your information to be displayed in a different window than the Applixware information.

HELP_SET_DEBUG@

Places Help into a debugging state

Format	HELP_SET_DEBUG@()
Description	Tells Help that it should write out internal state information while it is executing. This information includes the name of the target and the file in which the target resides (if the target is found).
See also	HELP_UNSET_DEBUG@

HELP_SET_ICON_TITLE@

Set's a window's icon text

Format	HELP_SET_ICON_TITLE@(domain, title)	
Arguments	domain	The name of a domain registered with Help using HELP_REGISTER@.
	title	The title that will appear below the Help window's icon.
Description	Sets the text of the icon when the window is minimized. This macro is usually used along with HELP_SET_APP_NAME@ so that the name of the window is the same (or similar to) the name in the window's title.	

HELP_SET_WINDOW_SIZE@

Sets the Help window's size

Format	HELP_SET_WINDOW_SIZE@(height, width)

Arguments	height	The Help window's height in mils (1000 mils = 1 inch). For example, 5000 is five inches.
	width	The Help window's width in mils.

Description Sets the height and width to be used when the Help window is *next* displayed. That is, if the Help window is displayed when you invoke this macro, the size of the window does not change. It will only be at this size if you close the window and reopen it.

See also **HELP_DEFAULT_WINDOW_SIZE@**

HELP_RESET_SIZE@

HELP_SORT_INDEX@

Creates the Help search list

Format format arrayof HELPindex@ items = HELP_SORT_INDEX@(indexText, topic, filenames, domains)

Arguments	indexText	A vector containing index terms.
	topic	A parallel vector containing the topics associated with an index term.
	filenames	A parallel vector containing the name of the file containing an index term.
	domains	A parallel vector containing the name of the domain of file containing an index term.

Description Returns the index used by Help's Search commands. This macro is called from within the bld_help.am file and is not meant to be used except by Help when it builds the search list.

HELP_UNFREEZE_DOMAIN@

Allows all domain's to be chosen

Format HELP_UNFREEZE_DOMAIN@()

Description Allows all domain's to be chosen after they have been locked out using **HELP_FREEZE_DOMAIN@**.

HELP_UNSET_DEBUG@

Ends Help's debugging state

Format HELP_UNSET_DEBUG@()

Description Tells Help that it should stop displaying debugging information.

See also **HELP_SET_DEBUG@**

HELP_UNREGISTER@

Removes a domain

Format HELP_UNREGISTER@(domain)

Arguments domain The domain being removed from help.

Description Removes knowledge of a domain from Help. After unregistering domain, you will no longer be able to access any of that's domain's information. If an entry from domain is in the history list, the user will receive an error when trying to access that entry.

There is seldom any reason to use this command except while debugging Help related macros.

See also **HELP_REGISTER@**

HOUR@

Extracts the hour value from a serial time number

Format HOUR@(timeNumber)

Arguments timeNumber A serial time number.

Description HOUR@ is a time function which extracts the hour (0-23) value from a serial time number. You can enter a serial time number as an argument for the HOUR@ function. A formula that contains the serial time number 0.427314815 (which represents a time of 10 hours, 15 minutes, and 20 seconds) returns 10.

You can also use the **NOW@** or **TIME@** function as arguments in the HOUR@ function. For example, the formula +HOUR@(TIME@(15,30,45)) returns 15.

HTML_.am

```
/* Names of WP styles/glossaries/macros etc. */
define          WP#HTML#HYPERLINK#MACRO
                        NXLT("WP_HTML_HYPERLINK@")
define          WP#HTML#GRAPHICS#EDIT#MACRO
                        NXLT("HTML_EDIT_FOREIGN_GRAPHICS@")
define          WP#HTML#H1#STYLE                NXLT("html_h1")
define          WP#HTML#H2#STYLE                NXLT("html_h2")
define          WP#HTML#H3#STYLE                NXLT("html_h3")
define          WP#HTML#H4#STYLE                NXLT("html_h4")
define          WP#HTML#H5#STYLE                NXLT("html_h5")
define          WP#HTML#H6#STYLE                NXLT("html_h6")
define          WP#HTML#BULLET#STYLE
                        NXLT("html_bullet_list")
define          WP#HTML#STARTNUMBER#STYLE
                        NXLT("html_num_list_start")
define          WP#HTML#NUMBER#STYLE
                        NXLT("html_num_list")
define          WP#HTML#DT#STYLE
                        NXLT("html_description_title")
define          WP#HTML#DD#STYLE
                        NXLT("html_description")
define          WP#HTML#PREFORMAT#STYLE
                        NXLT("html_preformatted")
define          WP#HTML#BLOCKQUOTE#STYLE
                        NXLT("html_blockquote")
define          WP#HTML#CAPTION#STYLE
                        NXLT("html_table_caption")
define          WP#HTML#HYPERLINK#STYLE
                        NXLT("html_hyperlink_text")
define          WP#HTML#HYPERLINK#WP#STYLE
                        NXLT("html_hyperlink_wp_text")
define          WP#HTML#EMPHASIS#STYLE
                        NXLT("html_emphasis_text")
define          WP#HTML#STRONG#STYLE
                        NXLT("html_strong_text")
define          WP#HTML#CODE#STYLE
                        NXLT("html_code_text")
define          WP#HTML#SAMP#STYLE
                        NXLT("html_sample_text")
define          WP#HTML#KBD#STYLE
```

```
                NXLT("html_kbd_text")
define      WP#HTML#VAR#STYLE
                NXLT("html_variable_text")
define      WP#HTML#DFN#STYLE
                NXLT("html_definition_text")
define      WP#HTML#CITE#STYLE
            NXLT("html_citation_text")
define      WP#HTML#ADDRESS#STYLE
                NXLT("html_address")
define      WP#HTML#UNKNOWN#STYLE
                NXLT("html_unknown_text")
define      WP#HTML#CELL#STYLE
                NXLT("html_cell")
define      WP#HTML#CELL#HEADER#STYLE
                NXLT("html_cell_header")
define      WP#HTML#HR#STYLE                    NXLT("html_hr")
define      WP#HTML#BULLET#GLOSSARY
                NXLT("html_bullet")
define      WP#HTML#NUMBER#GLOSSARY
                NXLT("html_numlist")
define      WP#HTML#STARTNUMBER#GLOSSARY
                NXLT("html_numliststart")
define      WP#HTML#TITLE#DOCVAR
                NXLT("HtmlTitle@")
define      WP#HTML#HEAD#COMMENTS#DOCVAR
                NXLT("HtmlHeadComments@")
define      WP#HTML#HEAD#TAGS#DOCVAR
                NXLT("HtmlHeadTags@")
define      WP#HTML#BODY#TAG#DATA#DOCVAR
                NXLT("HtmlBodyTagData@")
define      WP#HTML#AUTHORING#DOCVAR
                NXLT("HtmlAuthoringSystem$")
define      WP#HTML#HELP#BUILD#DOCVAR
                NXLT("HtmlHelpBuild$")
define      WP#HTML#BACKGROUND#DOCVAR
                NXLT("HtmlBackground@")
define      WP#HTML#WP#STYLES#DOCVAR
                NXLT("HtmlWpStyles@")
define      WP#HTML#CELL#WIDTH#DOCVAR
                NXLT("HTMLCellWidthPolicy@")
define      WP#HTML#ORIG#URL#DOCVAR
                NXLT("HTMLOriginalURL@")
define      WP#HTML#ORIG#PATH#DOCVAR
                NXLT("HTMLOriginalPath@")
```

```
define          WP#HTML#SERVER#URL#DOCVAR
                        NXLT("HTMLServerURL@")
define          WP#HTML#SERVER#ROOT#DOCVAR
                        NXLT("HTMLServerRoot@")
define          WP#HTML#BG#COLOR
                        NXLT("HtmlBackground@")
define          WP#HTML#TEXT#COLOR
                        NXLT("HtmlText@")
define          WP#HTML#LINK#COLOR
                        NXLT("HtmlLink@")
define          WP#HTML#LINK#DEFAULT#COLOR
                        NXLT("HtmlLinkDefault@")
define          WP#HTML#VLINK#COLOR
                        NXLT("HtmlVLink@")
define          WP#HTML#ALINK#COLOR
                        NXLT("HtmlALink@")

/* Here are all the possible HTML paragraph styles */
/*********
define          WP#HTML#PARA#STYLES
{
WP#HTML#H1#STYLE,
WP#HTML#H2#STYLE,
WP#HTML#H3#STYLE,
WP#HTML#H4#STYLE,
WP#HTML#H5#STYLE,
WP#HTML#H6#STYLE,
WP#HTML#BULLET#STYLE,
WP#HTML#STARTNUMBER#STYLE,
WP#HTML#NUMBER#STYLE,
WP#HTML#DT#STYLE,
WP#HTML#DD#STYLE,
WP#HTML#PREFORMAT#STYLE, WP#HTML#BLOCKQUOTE#STYLE,
WP#HTML#CAPTION#STYLE,
WP#HTML#CELL#STYLE,
WP#HTML#CELL#HEADER#STYLE,
WP#HTML#HR#STYLE }
*********/
define          WP#HTML#PARA#STYLES
{
WP#HTML#H1#STYLE,
WP#HTML#H2#STYLE,
WP#HTML#H3#STYLE,
WP#HTML#H4#STYLE,
WP#HTML#H5#STYLE,
WP#HTML#H6#STYLE,
WP#HTML#BULLET#STYLE,
WP#HTML#STARTNUMBER#STYLE,
WP#HTML#NUMBER#STYLE,
WP#HTML#DT#STYLE,
WP#HTML#DD#STYLE,
WP#HTML#PREFORMAT#STYLE, WP#HTML#BLOCKQUOTE#STYLE,
WP#HTML#CAPTION#STYLE,
WP#HTML#CELL#STYLE,
WP#HTML#CELL#HEADER#STYLE,
WP#HTML#HR#STYLE
}


/* Other defines needed in HTML Authoring System and import/export filters */

/* Table export policies */
define          HTML#CELL#ABS#WIDTH               0
/* Preserve cell widths on export */
define          HTML#CELL#REL#WIDTH               1
/* Preserve relative cell widths */
define          HTML#CELL#COLSPAN           2
/* Preserve colspan info only */



/*      HTML TreeNode Type */


/***
Any URL (absolute or relative) has the following format:
                <scheme>://<net_loc>/<path>;<params>?<query>#<fragment>

 which can be decomposed into the structure below. Note many of the
parts of this structure are optional.

 For more information, see R. Fielding's document "Relative Uniform
Resource Locators" at http://ds.internic.net/rfc/rfc1808.txt
***/

format www_url_parts@
        scheme,       ' Trailing ':' is clipped
        net_loc,      ' Leading '//' is clipped
        path,           ' Leading '/', if present, is NOT clipped
        params,         ' Leading ';' is clipped
        query,          ' Leading '?' is clipped
        fragment        ' Leading '#' is clipped

/***
The HTML Authoring System provides a user interface by which a user can specify many parameters
for the filtering of an Applixware Words file into HTML format (e.g. document title, text color, etc.).

The following structure allows the specification of the same parameters
when the filtering takes place via the FILTER_WP_TO_HTML@ macro
or the WP_EXPORT_HTML@ macro. This allows a high-quality conversion of Words documents in a
batch mode.
***/

/* Here are the defines and substructures used in the main structure */

define HTML#EXPORT#WINDOW#BG                    0
        ' Bring up BG window; export; shut down
define HTML#EXPORT#WINDOW#CURRENT               1
        ' Use caller's window
define HTML#EXPORT#WINDOW#BATCH                 2
        ' Use caller's window, but do not
                                                        ' pop up any dboxes

define HTML#EXPORT#GFX#NONE                     0
        ' Leave Ax Graphics insets alone
define HTML#EXPORT#GFX#GIF                      1
        ' Export Ax Graphics insets to GIF
define HTML#EXPORT#GFX#JPEG                     2
        ' Export Ax Graphics insets to JPEG

format html_export_behavior_styles@
        wp_style,       ' Name of a Words style; e.g. "Heading1"
        html_style      ' Corresponding HTML style to use; e.g. "html_h1".

'  The proper style names are defined above.
'  Note final stage of filtering to HTML looks for
'  these exact names.

format html_export_behavior_color@
        specified,              ' TRUE -> Good color value follows.
                                '  FALSE -> Defer to Web browser color
        red, green, blue        ' Values between 0 and 255
```

```
/* Here is the main structure */
format html_export_behavior@
window_mode,
          ' HTML#EXPORT#WINDOW#BG or #CURRENT; the window
          '  into which the Words document is imported,
          '  processed, and exported from.
          ' HTML#EXPORT#WINDOW#BATCH means current window,
          '  but 'batch' mode - no dboxes will pop up.
no_init,
          ' If TRUE, omit standard initializations (don't
          '  convert styles, charts, or links)
defaults,
          ' If TRUE, ignore all subsequent members of the
          '  structure and use default behavior (e.g.
          '  default title, no bg image...)
          '  Else behavior is specified below
format arrayof html_export_behavior_styles@ styles,
          ' This is a 2-D array matching WP & HTML styles.
          '  If absent, the default matchups are used for
          '  the Applixware-defined styles, but user-defined
          '  styles are not converted (assumed to be normal
          '  paras). If present, the default matchups are
          '  used for styles not specified in this array.
          title,            ' If present, the title of the HTML document.
                            '  If NULL, a default title will be provided.
          base_url,         ' If present, specify the Base URL of the doc.
                            '  If an empty string, the HTML doc will have
                            '  no Base URL
                    '  If NULL, defer to Base URL of WP or template doc,
                            '  if either specifies one.
          bg_image,         ' If present, the full pathname of an image to be
                    '  tiled across the background of the HTML document
                            '  (visible in a Web browser, such as Netscape).
                            '  The image should be in a standard Web format
                            '  (e.g. GIF), not Applixware Graphics.
                            '  If empty, the HTML document will have no
                            '  background image.
                            '  If NULL, defer to background image of WP or
                            '  template doc, if either specifies one.
          format html_export_behavior_color@ bg_color,
          format html_export_behavior_color@ text_color,
          format html_export_behavior_color@ link_color,
          format html_export_behavior_color@ vlink_color,
          format html_export_behavior_color@ alink_color,
                            ' These structures specify a color for the HTML
                            '  background, text, links, visited links, and
                            '  active links respectively.
                            '  For each structure, if the 'specified' flag is
                            '  TRUE, a good color is given
                            '  If the 'specified' flag is FALSE, this means
                            '  don't specify the color; allow the Web browser
                            '  to use its own defaults.
                            ' If the structure is NULL, defer to the color in
                            '  the WP or template doc, if either specifies one.
                            '  comments,
                            ' An array of strings to be placed in the HEAD
                            '  section of the HTML document as comments.
                            '  Each member of the array is made a separate
                            '  comment.
                            ' Leave NULL for no comments.
          gfx_export,
                    ' If HTML#EXPORT#GFX#NONE, leave Ax Graphics
                    '  insets alone.
                    '  If HTML#EXPORT#GFX#GIF or
                    '     HTML#EXPORT#GFX#JPEG,
                    '  export to GIF or JPEG respectively, updating
                    '  the inset field to link to the new foreign file.
                    '  If NULL, use the current preference value
          gfx_directory, ' Only meaningful if creating GIF/JPEG files for
                            '  Applixware Graphics insets. This is an absolute
                            '  or relative (with respect to the HTML file being
                            '  created) location for the new GIF/JPEG files.
                            '  If empty string, the GIF/JPEG files are put in
                            '  the same directory as the HTML file.
                            '  If NULL, use the current preference value
          gif_trans_inter,        ' Only meaningful if creating GIF files for
                            '  Applixware Graphics insets.
                            '  If TRUE, the GIF89 filter is used, and thus
                            '  transparency in the graphics (if any) is
                            '  preserved, and the graphics is interlaced.
                            '  If FALSE, the GIF87 filter is used, and thus
                            '  transparent colors are no longer transparent
                            '  (probably will display as white, but could be
                            '  another color - see Graphics doc for more info),
                            '  and the graphics is not interlaced.
                            '  If NULL, use the current preference value
          table_borders,
                            ' If TRUE, all tables in the HTML document will be
                            '  given borders.
                            '  If FALSE, no tables will have borders.
                            '  If NULL, use the default behavior: for each
                            '  table, the number of borders the upper left cell
                            '  has determines whether the table has borders.
          cell_width,
                    ' One of HTML#CELL#ABS#WIDTH, REL#WIDTH, or
                    '  COLSPAN. Determines how strictly cell widths
                    '  are preserved for all tables.
                    ' If NULL, use the default behavior
          no_cvt_charts,
                    ' If FALSE, convert Applixware charts to GIF's
                    ' If TRUE, leave charts alone
                    ' If NULL, use default behavior (convert)
          no_cvt_links
                    ' If FALSE, convert Applixware hyperlinks into
                    '  HTML-style hyperlinks
                    ' If TRUE, leave hyperlinks alone (filter will not
                    '  convert them into HTML links properly)
                    ' If NULL, use default behavior (convert)
```

HTMLX_ASSIGN_INSTANCE@

Assigns an HTML Author handle
to an Applixware Task

Format HTMLX_ASSIGN_INSTANCE@(HTMLX, taskid)

Arguments

HTMLX	An Applixware HTML Author handle as returned by HTMLX_CREATE_INSTANCE@.
taskId	An Applix*ware* task ID number.

Description Assigns the handle indicated by HTMLX to task taskid. This macro changes the task's owner from its current task to a different task.

The HTMLX handle changes during the execution of the macro. The original handle is set by the programmer on input. The new handle is returned to the calling program after the successful completion of the call.

HTMLX_COPY_INSTANCE@

Creates a Copy of an HTML Author handle

Format HTMLX_COPY_INSTANCE@(HTMLX, taskid, newHTMLX)

Arguments

HTMLX	An Applixware HTML author handle as returned by HTMLX_CREATE_INSTANCE@.
taskId	An Applix*ware* task ID number.
newHTMLX	A copy of the HTMLX handle. On input, set this argument to NULL.

Description Creates a copy of the Applixware HTML Author handle HTMLX. The HTMLXNew argument becomes an exact copy of the HTMLX argument.

HTMLX_CREATE_INSTANCE@

Creates an instance of the Applixware
HTML Author in memory

Format HTMLX_CREATE_INSTANCE@(HTMLX, uid)

Arguments

HTMLX	The Applixware HTML Author handle to create. On input, set this variable to NULL. After HTMLX_CREATE_INSTANCE@, this variable contains an Applixware HTML Author handle.
uid	An Applixware task ID. This is an optional value indicating the task to which the HTML Author handle is assigned. If this argument is omitted or set to zero (0), the current task is used.

Description Creates an internal instance of the Applixware Author. A *handle* to this internal editor is returned. You can associate an Applixware HTML file with this handle (HTMLX_READ_FILE@) and display the Applixware Words file as an inset in a dialog box (DB_CTRL_INSET@).

HTMLX_DESTROY_INSTANCE@

Format HTMLX_DESTROY_INSTANCE@(HTMLX)

Arguments

HTMLX	An Applixware HTML author handle to destroy.

Description Frees all the resources associated with a HTMLX handle. After running this macro, you can no longer use the handle to manipulate the HTML Author object.

HTMLX_GET_INSTANCE@

Returns a task's HTML Author handle

Format HTMLX_GET_INSTANCE@(HTMLX, taskID)

Arguments

HTMLX	The Applixware HTML Author handle to create. On input, set this variable to NULL.
uid	An Applixware task ID. This is an optional value indicating the task to which the HTML Author handle is assigned. If this argument is omitted or set to zero (0), the current task is used.

Description Returns the HTML Author handle associated with an Applixware task.

HTMLX_READ_BUFFER@

Associates a data buffer
with an HTML Author handle

Format HTMLX_READ_BUFFER@(HTMLX, buffer)

Arguments

HTMLX	an Applixware HTML Author handle as returned by HTMLX_CREATE_INSTANCE@.
buffer	A buffer containing the data from a Words document.

Description Reads the contents of a buffer and associates the data with an HTML Author handle.

HTMLX_READ_FILE@

Reads an HTML Author file and assoociates the
data with a handle

Format HTMLX_READ_FILE@(HTMLX, filename)

Arguments

HTMLX	An Applixware HTML Author handle as returned by the macro HTMLX_CREATE_INSTANCE@.
filename	The name of the file being read.

Description Reads the contents of a HTML file and assigns it to the handle HTMLX.

While the file is being read, HTMLX_READ_FILE@ places a read lock on the file. This lock is cleared when the file read is complete. If two ELF macros try to read the same file simultanously, one of the ELF macros will throw an error indicating that the file is locked.

HTMLX_WRITE_BUFFER@

Writes data associated with an HTML author handle to a memory buffer

Format HTMLX_WRITE_BUFFER@(HTMLX, buffer)

Arguments

HTMLX	an Applixware HTML Author handle as returned by HTMLX_CREATE_INSTANCE@.
buffer	An initialized ELF variable

Description Writes the data associated with the handle HTMLX to a buffer.

HTMLX_WRITE_FILE@

Reads an HTML Author file and associates the data with a handle

Format HTMLX_WRITE_FILE@(HTMLX, filename)

Arguments

HTMLX	An Applixware HTML Author handle as returned by the macro HTMLX_CREATE_INSTANCE@.
filename	The absolute pathname of a file.

Description Writes the data associated with an HTML author handle to a file.

While the file is being written, HTMLX_WRITE_FILE@ places a write lock on the file. This lock is cleared when the file write is complete. If two ELF macros try to write the same file simultanously, one of the ELF macros will throw an error indicating that the file is locked.

HTML_APPLICATION_DLG@

Creates a new HTML Author window

Format taskID = HTML_APPLICATION_DLG@ ([menubarID][,windowlessFlag][,hooklessFlag])

Arguments

menubar_id The number of a menu bar to be associated with this window. (This argument is optional.) This number should be a number between 200 and 299. If the value is set to NULL, you will receive the default menu bar.

windowlessFlag
Boolean value where TRUE indicates that no window will be displayed. FALSE is the default.

hooklessFlag
A Boolean value where TRUE indicates that no hook macro is run at startup time. If the value is FALSE, HTML Author runs the hook macro configured in the Words preferences.

Description Creates a new HTML Author window. This window can use the default menu bar or it can use the menu bar associated with menubar_id. The task id for the newly created task is returned.

The optional menubar_id parameters lets you load a menu bar according to the task that will be performed in the HTML Author window. You could even display several versions at the same time by opening several windows, each with separate numbers. To display a custom menu bar, you have to first load it into memory using **SET_SELECTIONS@**.

The windowlessFlag parameter lets you perform automated tasks "in the background," without displaying the Words application window. Using this option, two applications can work simultaneously without interrupting each other and without invoking two separate axmain processes. This windowless Words application becomes a "child" of the window from which it was invoked.

Performing non-interactive tasks without a window conserves computing resources as the window will never have to be displayed.

To perform a windowless task:

- You must suppress all messages that are displayed. Otherwise, the task will hang when it tries to display them. To suppress information displayed by INFO_MESSAGE@, use **SUPPRESS_INFO_MESSAGES@**. Suppress error messages with the **SUPPRESS_ERROR_MESSAGES@** macro.
- You cannot include any prompts using PROMPT@.

- Be sure to explicitly exit any windowless Words applications after they are complete. Otherwise, the tasks remain in memory until you log out.
- It is best to make a call to SELECT_WINDOW@ when exiting the windowless application. In this way, you guarantee that the exit command is invoked against the right window.
- Before testing a newly written macro that invokes a windowless application, it is best to first test the macro with all windows displaying in the foreground.

The hooklessFlag parameter determines whether a hook macro is run when the HTML Author application starts. A hook macro is configured through the Words preferences dialog, and runs whenever you start HTML Author or open an HTML file.

If you configure a hook macro to run at startup time, and you set hooklessFlag to NULL, the hook macro runs. If you configure a hook macro to run at startup time, and you set hooklessFlag to TRUE, the hook macro does not run. This is an optional parameter.

HTML_APPLICATION_DLG@ is called by the * ® HTML Author menu option.

HTML_NEW@

Opens a new HTML document

Format HTML_NEW@()

Description Opens a new HTML document with the Applixware HTML Author. The default HTML template is used.

HTML_NEW_WINDOW@

Opens an HTML document in a new window

Format HTML_NEW_WINDOW@([id], [quiet], [readonly], [width], [height], [windowtitle], [icontitle], [iconid])

Arguments

menubar_id	The number of a menu bar to be associated with this window. (This argument is optional.) This number should be a number between 200 and 299. If the value is set to NULL, the window is assigned the default menu bar.
quiet	Boolean value where TRUE indicates that no window will be displayed. FALSE indicates that a normal HTML window is displayed.
readonly	A Boolean value where TRUE indicates that the window is read only. If the value is FALSE, the window can be edited.

width	The width of the window in mils (1000 mils = 1 inch). If this option is omitted, the system default is used for the window width.
height	The height of the window in mils. If this option is omitted, the system default is used for the window height.
windowtitle	The title displayed at the top of the window. If this option is omitted, the name of the file is used as the window title.
icontitle	The title displayed below the icon when the window is minimized. If this option is omitted, the name of the file is used as the icon title.
iconId	The ID of the icon used when the window is minimized. If this option is omitted, the default HTML icon is used.

Description Opens a new HTML document in a new window. Using the arguments, you can change the parameters of the window, such as its size and title.

HTML_OPEN_HTML@

Open an HTML document with the HTML Author

Format HTML_OPEN_HTML@(filename, revert, url)

Arguments

filename	The name of an HTML file.
revert	Indicates whether to place the imported document in the current HTML window or a new HTML window. If revert is set to TRUE, the imported document replaces the current document. The current document is not saved. If set to FALSE, a new HTML window is created and the imported file is placed in the new window.
URL	The file's complete URL if it was accessed using the World Wide Web. This URL is used to resolve any relative URLs contained within filename.

HTML_OPEN_URL@

Open a URL with the HTML Author

Format HTML_OPEN_URL@(url)

Arguments

URL	The file's complete URL if it was accessed using the World Wide Web. This URL is used to resolve any relative URLs contained within filename.

Description Imports an HTML file from its location on the World Wide Web into the HTML Author. The file is opened as a temporary file on the local file system.

HTML_OPEN_WP@

Open an Applixware Words document with the HTML Author

Format HTML_OPEN_WP(Filename)

Arguments

Filename	Absolute pathname of an Applixware Words document to open with the HTML Author.

Description Imports an Applixware Words file into the HTML Author.

HTML_QUICK_BULLETS@

Adds a bullet to text

Format HTML_QUICK_BULLETS@()

Description Applies the paragraph style html_bullet_list to the current text.

HTML_QUICK_NUMBERING@

Adds a number to text

Format HTML_QUICK_NUMBERING@()

Description Applies the paragraph style html_num_list to the current text.

HTML_SAVE@

Saves an HTML file

Format HTML_SAVE@(oldName, newName, format html_export_behavior@ behavior)

Arguments

oldName	A string containing the original name of the document to be saved
newName	A string containing the new name of the document to be saved

behavior	An ELF html_export_behavior@ format. This format is described in the file html_.am.

Description Saves the current Applixware Words document as an HTML file. If the oldName and newName parameters contain the same string, the Words file is overwritten by an HTML file of the same name.

HTML_TOGGLE_RULER_DISPLAY@

Turn the ruler on and off in the HTML Author

Format HTML_TOGGLE_RULER_DISPLAY@()

Description Toggles the ruler on and off in the HTML Author.

HTML_TOGGLE_VIEW_MODE@

Turn View Mode on and off

Format HTML_TOGGLE_VIEW_MODE@()

Description By default, the HTML Author starts in Edit Mode. This mode is designed for editing HTML documents. In this mode, double-clicking on a hyperlink brings up the Edit Hyperlink dialog box rather than executing the hyperlink. In View Mode, double-clicking on a hyperlink executes the hyperlink.

Use this macro to turn View Mode on and off.

HTML_VIEW_EXPRESSLINE@

Turn HTML Author Expressline on and off

Format HTML_VIEW_EXPRESSLINE@()

Description Toggles the HTML Author *Express*line on and off.

INFO_FROM_FILE_HEADER@

Returns a format based on a document's file header

Format format afile_info info = INFO_FROM_FILE_HEADER@(fileHeader)

Arguments fileHeader The file header contained within an Applix*ware* document.

Description Returns a afile_info format that describes the information contained within an Applix*ware* file header. The definition of this format is as follows:

```
format afile_info
    encoding, 'TRUE if file is encoded
    version,  'version number of current format
    docType,  'as in regcfil_.am
              'version number of original document
    original_version,
              'last version capable of reading this file
    minimum_version,
              'arbitrary hint string
    content_hint
```

INFO_MESSAGE@

Displays an information dialog box

Format INFO_MESSAGE@(message)

Arguments message A string representing the message to be displayed in the dialog box. The newline character, \n, may be included in message to display text on multiple lines.

Description Displays a message dialog box with the message you specify. Information message boxes supply information about an application to a user. The information message box includes an OK button. The INFO_MESSAGE@ string is limited to 507 characters.

ELF prevents this macro from displaying the same message twice in a row.

Example

INSTALL_AXNET_LIBRARY@

Installs the shared libraries registered with an axnet service

Format INSTALL_AXNET_LIBRARY@(hostname, service[, useHourglassFlag])

Arguments

hostname	The name of the machine upon which the service resides.
service	The name of the service as it was registered with **AXNET_SERVICE_REGISTER@**.
useHourglassFlag	A Boolean value which if set to TRUE indicates that an hourglass is displayed. The default is FALSE. See **RPC_CHANNEL_USE_HOURGLASS@** for more information.

Description Installs the C functions contained within a shared library registered with an axnet service.

INSTALL_C_LIBRARY@

Adds a shared library of C functions

Format INSTALL_C_LIBRARY@(pathname[, funcNames])

Arguments

pathname	The full path name of the shared library file.
funcNames	An array containing the names of the C functions contained within pathname.

Description Binds in a shared C library, declaring the C functions in this library. This macro simply defines functions names and where they exist. The binding of the shared library to Applix*ware* occurs when a macro invokes one of the functions that exist in the shared library.

If the optional funcNames argument is omitted, your C program must contain a function named AxGetCallInfo whose purpose is to name the functions contained within the C library. If you ae creating functions that should be listed in the Spreadsheets function box, you must use create this C function.

The purpose of the AxGetCallInfo function is return information about the functions. Here is an example.

```
/* call info table; this is defined in elfapi.h */
static AxCallInfo_t funcTable[] = {
        { "General", MULTNUM, "MULTNUM", "MULTNUM(number1,number2)", TRUE},
        { "General", ADDNUM, "ADDNUM", "ADDNUM(number1, number2)", TRUE},
        { NULL, NULL, NULL, NULL, FALSE}        /* terminator */
};

AxCallInfo_t *AxGetCallInfo_ud_1()
{
        return(funcTable);
}
```

INSTALL_DOC@

Loads and compiles a macro document

Format INSTALL_DOC@(filename)

Arguments filename — The full path name of the macro document. The file name does not include the .am suffix.

Description Loads and compiles all macros in an ELF source document.

See also INSTALL_FILE@
INSTALLED@

INSTALL_FILE@

Loads and compiles a macro document

Format INSTALL_FILE@(filename)

Arguments filename — The full path name of the macro document.

Description Loads and compiles all macros in the specified Macro Editor document.

See also INSTALL_DOC@
INSTALLED@

INSTALL_PROGRAM_FILE@

Loads program information into ELF memory

Format INSTALL_PROGRAM_FILE@(filename)

Arguments	filename	The name of the file containing macros and dialog boxes in ELF machine-independent format. By convention, these files end with the extension ".prg". However, this convention is not enforced.

Description Loads the macros and dialog boxes contained within filename and makes them ready for execution.

NOTE: INSTALL_PRG@ is a synonym for this macro.

NOTE: Program files can only be created by Applixware at this time. However, they can be unpacked by any one who uses this macro.

INSTALLED@

Returns TRUE if a macro is installed

Format flag = INSTALLED@(name)

Arguments	name	The name of the macro whose existence is being tested.

See also INSTALL_DOC@
INSTALL_FILE@

INT@

Returns integer equivalent to an expression

Format num = INT@(value)

Arguments	value	A numeric expression

Description Truncates value at its decimal point. It does not round value up or down.

Example

See also ABS@

IS_ARRAY@

Determines if an argument is an array

Format flag = IS_ARRAY@(value)

Arguments value The value you wish to evaluate.

Description Returns TRUE if value is an array. Returns FALSE if value is not an array. If the value of the argument is NULL, IS_ARRAY@ returns FALSE. If the value of the argument is an array having zero elements, IS_ARRAY@ returns TRUE.

See also IS_BINARY@
IS_NULL@
IS_NUMBER@
IS_NUMERIC_STRING@
IS_STRING@
IS_WHITESPACE@

IS_ASCII_FILE@

Determines if a file is ASCII

Format flag = IS_ASCII_FILE@(filename)

Arguments filename The file you are checking to see if it is ASCII.

Description Returns TRUE if filename is a seven-bit ASCII file. Returns FALSE otherwise.

See also IS_FILE_OPEN@
IS_UUENCODED@

Example

IS_BINARY@

Determines if an argument is a binary data object

Format flag = IS_BINARY@(value)

Arguments	value	The value you wish to evaluate.

Description Returns TRUE if value is a binary object. Returns FALSE if value is not a binary object.

See also IS_ARRAY@
IS_NULL@
IS_NUMBER@
IS_NUMERIC_STRING@
IS_STRING@
IS_WHITESPACE@

IS_ELFRT@

Determines whether you are running axmain or ELFRT

Format flag = IS_ELFRT@()

Description Returns TRUE if you are running the stand-alone version of ELF. (This is used with unbundled ELF, Applixware Mail, and Applixware Data when the base product is not also purchased.) If you are running from axmain, FALSE is returned.

IS_FACE_LATIN@

Returns TRUE if font contains Latin characters

Format flag = IS_FACE_LATIN@(fontName)

Arguments	fontName	The name of one of the fonts returned by LIST_FONT_FAMILIES@.

Description Returns TRUE if the font is a Latin font. (A *Latin* font is one that contains the standard alphabetic and numeric characters.) This macro returns TRUE for all of the Applix*ware* fonts except for the Symbol and Dingbats fonts.

IS_FILE_OPEN@

Indicates if a file is open

Format flag = IS_FILE_OPEN@(filename)

Arguments

filename	The name of the file being checked.

Description Returns TRUE is the file is already open by another ELF or Applix*ware* task. Returns FALSE if the file is not open or if the file does not exist.

See also IS_ASCII_FILE@
IS_UUENCODED@

IS_NULL@

Determines if a value is null

Format flag = IS_NULL@(value)

Arguments

value	The value you wish to evaluate.

Description Returns TRUE if value is NULL. It returns FALSE if value is not NULL.

The value of every uninitialized variable is NULL. You can also explicitly set an already initialized variable to NULL.

See also IS_ARRAY@
IS_BINARY@
IS_NUMBER@
IS_NUMERIC_STRING@
IS_STRING@
IS_WHITESPACE@

IS_NUMBER@

Determines if a value is a number

Format flag = IS_NUMBER@(value)

Arguments	value	The value you wish to evaluate.

Description Returns TRUE if value is a number; it returns FALSE if value is not a number.

See also IS_ARRAY@
IS_BINARY@
IS_NULL@
IS_NUMERIC_STRING@
IS_STRING@
IS_WHITESPACE@

IS_NUMERIC_STRING@

Determines whether a passed string is numeric

Format flag = IS_NUMERIC_STRING@(string)

Arguments	string	The passed string.

Description Determines whether string is numeric. For example, "-34" is and "abc" is not numeric.

See also IS_ARRAY@
IS_BINARY@
IS_NULL@
IS_NUMBER@
IS_STRING@
IS_WHITESPACE@

IS_ORIENT@

Returns TRUE if in the Applix*ware* oriental environment

Format flag = IS_ORIENT@()

Description Returns TRUE is you are running in the Applix*ware* oriental environment. If you are not in this environment, FALSE is returned.

IS_STRING@

Determines if a value is a string

Format flag = IS_STRING@(value)

Arguments value The value you wish to evaluate.

Description Returns TRUE if value is a string; it returns FALSE if value is not a string.

See also IS_ARRAY@
IS_BINARY@
IS_NULL@
IS_NUMBER@
IS_NUMERIC_STRING@
IS_WHITESPACE@

IS_UUENCODED@

Indicates whether or not a file has been uuencoded

Format flag = IS_UUENCODED@(filename)

Arguments filename The path name of the file to be checked for *uuencoding*.

Description Returns TRUE if a file was uuencoded, FALSE if it was not uuencoded. *Uuencoding* converts a file into an ASCII-encoded representation that can be sent using Mail.

See also IS_ASCII_FILE@
IS_FILE_OPEN@

IS_WHITESPACE@

Determines if a character is a blank or a tab

Format flag = IS_WHTESPACE@(char)

Arguments char The character being evaluated.

Description	Returns TRUE if char is blank or a tab.
See also	IS_ARRAY@ IS_BINARY@ IS_NULL@ IS_NUMBER@ IS_NUMERIC_STRING@ IS_STRING@

KILL_TASK@

Stops an ELF or Applix*ware* task

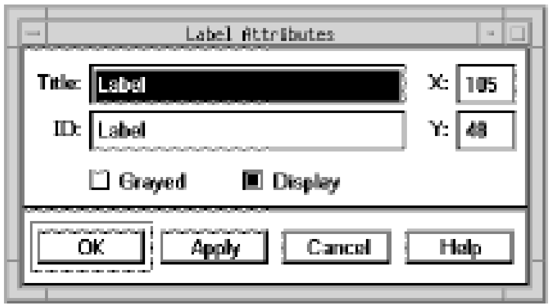
Format	KILL_TASK@(id)	
Arguments	id	The unique ID of the task you want to stop.
Description	Terminates the task having the ID you specify. Processing does not continue until the task is eliminated. You can use TASK_LIST@ to obtain the ID of a task.	
See also	TASK_LIST@	

Label Control

A label is a control that has no programmable features. Its purpose is to display a text string in a dialog box. To add a label to a dialog box, choose Controls Ý Label, and click in the dialog box.

Label Attributes

When you double-click a label, the Label Attributes dialog appears:



The Label Attributes dialog allows you to set the following attributes:

Display	Turn on Display to make the label display in the dialog box. Turn off Display to hide the label.
Grayed	Turn on Grayed to make the label grayed. Turn off Grayed to make the label display normally.
Title	Enter the string that you want the label to display in the Title field.

LEN@

Returns a string's length

Format length = LEN@(text)

Arguments text — The string whose length is returned

Description Returns the number that specified the amount of characters in a text string. If text is empty (""), LEN@ returns the value 0.

Example

LIST_FIND_ALPHA@

Finds a word in a list of words

Format index = LIST_FIND_ALPHA@(wordArray, word)

Arguments

wordArray	An array of words.
word	The word being checked.

Description Searches for word within wordArray and returns its index if it is found. Note, that word-Array must be in alphabetic order (case insensitive) and the match must be exact. If the word is not found, "-1" is returned.

Example

See also [ARRAY_INDEX@]
[LIST_INSERT_ALPHA@]
[LIST_REMOVE@]

LIST_FONT_FAMILIES@

Returns an array of Applix*ware* font families

Format fontNameArray = LIST_FONT_FAMILIES@()

Description Returns a string array whose elements are the names of all available Applix*ware* fonts.

LIST_IMAGES@

Returns a list of all Glom file image names

Format imageList = LIST_IMAGES@( )

Description Returns a list of all the bitmap images contained within a special compressed area of Applix*ware* known as the *glom*.

LIST_INSERT_ALPHA@

Inserts a word into a list of words

Format newArray = LIST_INSERT_ALPHA@(wordArray, word)

Arguments

wordArray	An array of words.
word	The word being inserted into the array.

Description Inserts word in alphabetic order (case insensitive) into wordArray. wordArray must be in alphabetical order. The updated list is returned.

See also ARRAY_INSERT@

LIST_FIND_ALPHA@
LIST_REMOVE@

LIST_OF_DIRS@

Returns a list of directories

Format dirArray = LIST_OF_DIRS@(dir)

Arguments dir The directory from where you want a list.

Description Returns an array of directory names that do not have "." as a leading character in the directory name.

Example

See also FILE_SYSTEM_DIRS@
LIST_OF_FILES@

LIST_OF_FILES@

Returns filenames in a directory

Format fileArray = LIST_OF_FILES@(path)

Arguments path The path of the directory whose files are of interest.

Description Returns a string array containing the names of files contained within the specified directory.

Example

See also FILE_SYSTEM_FILES@
FILE_SYSTEM_INFO@
LIST_OF_DIRS@

LIST_OF_PRINTERS@

Returns a list of available printers

Format printerArray = LIST_OF_PRINTERS@()

Description Returns a string array in which each string element is a printer name defined for the system. If no printers are defined, NULL is returned. LIST_OF_PRINTERS@ looks in /etc/passwd for printers. If an override file exists, LIST_OF_PRINTERS@ will use the printers in it. All Applix*ware* Print dialog boxes use LIST_OF_PRINTERS@ to get the printer list.

Example

LIST_OF_WINDOWS@

Returns an array containing all windows

Format format arrayof window_format@ = LIST_OF_WINDOWS@()

Description Returns an array containing information about each window. Each element in the array consists of one window_format@ format. The definition of this format is as follows:

```
format window_format@
        id,         ' The large window id. This is a number greater than 100000
        title,      ' The window's title line
        icon_id,    'The icon number
        task_id,    'The task number of the window's owner
        menubar_id
                    'The window's menubar id number.
```

Example

LIST_OF_WINDOW_TITLES@

Returns an array of all window titles

Format titles = LIST_OF_WINDOW_TITLES@()

Description Returns an array titles containing the window titles of all Applix*ware* windows for the current user.

LIST_REMOVE@

Removes a word from a list of words

Format newArray = LIST_REMOVE@(wordArray, index)

Arguments

wordArray	The list of words you are removing a word from.
index	The place where you want the word removed in word_list.

Description Given an index and a list of words, LIST_REMOVE@ removes the word at index from the word_list. Then it closes up the list, and returns it to the user. A bad index value is ignored.

See also ARRAY_DELETE@
LIST_FIND_ALPHA@
LIST_INSERT_ALPHA@
SUBARRAY_REMOVE@

LIST_SCREENFONTS@

Returns a list of available screen fonts

Format nameArray = LIST_SCREENFONTS@()

Description Returns a list of the fonts that can be used to display information. These are the fonts provided by the server. It is not the list that is displayed when you are in applications such as Words on the Express*Line* pulldown list.

LOAD_GRAPHIC@

Attaches the graphic to a task

Format LOAD_GRAPHIC@(gfx[, taskID])

Arguments

gfx	A graphics handle.
taskID	The task to which the graphics editor will be attached.

Description Attaches the graphic to a task. If the task is not named, the graphic is assigned to the main Applix*ware* task.

See also ASSIGN_GRAPHIC@

CREATE_GRAPHIC@

DESTROY_GRAPHIC@

GET_GRAPHIC@

READ_GRAPHIC_FILE@

LOCALIZE_LINKS@

Localizes a document's links

Format LOCALIZE_LINKS@(filename, saveFile, parentDoc)

Arguments

filename	The file name containing the links being localized.
saveFile	The file that will contain the new information. That is, after the links are localized, this is the place to which the new file is written.
parentDoc	The name of the document's parent document and is used to get the original path (for resolving this document's links). This argument is useful if you are localizing a document's links from a copy of the original file that is in a different directory, or if you're localizing links of an embedded object.

Description Localizes all of a document's links. *Localizing* means that the link object's data is brought into the document. It does not mean that the document will be converted from one format to another. For example, an external ASCII file is not converted into an Applixware Words document when the link to it is localized.

This macro is slower than LOCALIZE_LINKS_FROM_INFO@ in that it works with a flattened version of the link information.

LOCALIZE_LINKS_FROM_INFO@

Localizes a document's links

Format LOCALIZE_LINKS_FROM_INFO@(filename, format doc_links_info@ info, saveFile, parentDoc)

Arguments

filename	The file name containing the links being localized.

info	The array of recursive link format information returned by **GET_LINKS_INFO@**.
saveFile	The file that will contain the new information. That is, after the links are localized, this is the place to which the new file is written.
parentDoc	The name of the document's parent document and is used to get the original path (for resolving this document's links). This argument is useful if you are localizing a document's links from a copy of the original file that is in a different directory, or if you're localizing links of an embedded object.

Description Localizes all of a document's links. *Localizing* means that the link object's data is brought into the document. It does not mean that the document will be converted from one format to another. For example, an external ASCII file is not converted into an Applixware Words document when the link to it is localized.

This macro is faster than **LOCALIZE_LINKS@** in that it works with a recursive version of the link information.

LOG@

Returns the natural logarithm

Format num = LOG@(value)

Arguments value A numeric expression greater than 0.

Description Returns the natural logarithm (base *e*) of a numeric expression.

Example

See also **COS@**
LOG10@
SIN@
SQR@

LOG10@

Returns the base 10 logarithm

Format num = LOG10@(value)

Arguments value A numeric expression greater than 0.

Description Returns the base 10 logarithm of a numeric expression.

Example

See also COS@
LOG@
SIN@

LOGOUT@

Logs out of Applix*ware*

Format LOGOUT@([exitCode])

Arguments	exitCode	A numeric value that is sent to the shell as exit code for this Applix*ware* process.

Description LOGOUT@ closes all Applix*ware* windows and exits Applix*ware*. LOGOUT@ does not save documents before closing Applix*ware* windows. Make sure you have saved all documents before calling LOGOUT@.

Example

See also EXIT_ALL@

LOWERCASE@

Changes text into lowercase

Format newText = LOWERCASE@(text)

Arguments	text	An alphanumeric string.

Description Returns transformed text that has all uppercase letters changed into lowercase. Strings in ELF are case sensitive, so text strings must be converted to the same case if you want to compare them.

See also UPPERCASE@

MACHINE_TYPE@

Returns the hardware platform type

Format id = MACHINE_TYPE@()

Description Returns a number indicating the type of hardware upon which Applix*ware* is executing. Possible values are:

1	DEC ULTRIX
2 (and **SYS5@**())	SUN4 and Solaris
3	HP300
4	SUN3 (no longer used)
5	MIPS
6	SCO UNIX
7	SGI
8	HP 800
9	RIOS
10	AUX
11	VAX
12	M_88K
13	CLIX Intergraph
14	AXMS
15	Alpha
16	USL5
17	SUNOS
18	LINUX

Example

MACRO_PARENT_APPID@

Returns the application ID

Format appID = MACRO_PARENT_APPID@()

Description Returns the application ID (as defined in app_ids.am) of the caller's parent task. If the task's parent is not a C based application, NULL is returned.

For example, if the caller's parent is Spreadsheets, 2 (which is APP#SPREADSHEET_) is returned.

MACRO_PARENT_KEEP_BLOCKED@

Indicates if a task should be unblocked

Format MACRO_PARENT_KEEP_BLOCKED@(flag)

Arguments

flag	A Boolean value which if passed a value of FALSE, immediately unblocks a task. TRUE means block it (or keep it blocked).

Description Controls the unblocking of a task that has executed a **NEW_TASK@**. For example, a child task can unblock its parent task before it normally would be ready to unblock. (This macro can only be used by a child task to unblock its parent.)

If flag is TRUE and the parent task is not unblocked, this macro keeps the parent from getting unblocked if such macros as DELAY@, DB_DISPLAY@, or RPC_XXX@ are used.

MACRO_PARENT_TASK@

Determines what task owns this task

Format taskID =MACRO_PARENT_TASK@()

Description Returns the task id of the task that spawned this task.

See also **ELF_PARENT_TASK@**

ELF_TASK_ID@

MACRO_WINS_BUSY@

Displays a busy indicator for task's windows

Format MACRO_WINS_BUSY@( )

Description Sets the hourglass cursor indicating that the task is busy for all of a task's windows. ELF changes this hourglass cursor back to a normal cursor when the task is ready to receive input. (The task indicates it is ready to receive input by executing such macros as **DB_DISPLAY@**, **GET_MESSAGE@**, and **DELAY@**.)

This macro differs from ALL_WINDOWS_BUSY@ in that it lets you set the hourglass at the task level rather than at the Applix*ware* or ELF level.

See also ALL_WINDOWS_BUSY@
RPC_CHANNEL_USE_HOURGLASS@

MAIN_DBOX@

Displays the Applix*ware* main menu

Format MAIN_DBOX@()

Description Displays the Applix*ware* main menu and makes it active.

MAX@

Returns the greater of the two numbers specified

Format maxValue = MAX@(a,b)

Arguments

a	A numeric value.
b	A numeric value.

Description Returns the greater of the two numbers a or b. You can use scientific notation when specifying these numbers.

Example

See also MIN@

MAX_STRING_LENGTH@

Returns the size of the longest array element

Format size = MAX_STRING_LENGTH@(array)

Arguments

array	An array name.

Description Returns the string length of the longest element in a one-dimensional array.

MENU_BAR_DLG@

Displays the Menu Bar Editor

Format MENU_BAR_DLG@(id, name)

Arguments

id	A unique number identifying the menu definition. id can be in the range of 21 to 99.
name	The menu bar definition file name. To create a new menu bar file, supply a new, unique name. To edit an existing file, supply the existing name. All menu bar definition files are assumed to be in the user's axhome directory. Therefore, do not provide a path name with name. New menu bar definition files will be saved in the user's axhome directory.

Description Displays the Applixware Dialog Box Editor. This function is useful for creating menus to be used in dialog boxes. If desired, you can copy an existing menu bar definition file and specify the copy as name. You can then use the menu bar editor to edit the menu to fit your needs.

See also SET_SELECTIONS@

UPDATE_SELECTIONS_FILE@

METRIC@

Indicates if Applix*ware* is running in metric mode

Format flag =METRIC@()

Description Returns TRUE if Applix*ware* is running in metric mode.

Example

See also METRICSYSTEM@.

METRICSYSTEM@

Indicates whether measurements display in metric units or inches

Format flag = METRICSYSTEM@()

Description METRICSYSTEM@ returns TRUE if a user's **Use Metric Measurement Units** preference in the Applixware Preferences is set to metric measurements, and FALSE if it is set to measure in inches.

See also METRIC@.

MIN@

Returns the lesser of the two numbers specified

Format minValue = MIN@(a,b)

Arguments		
	a	A numeric value.
	b	A second numeric value.

Description Returns the lesser of the two numbers specified. You can use scientific notation.

Example

See also MAX@

MINUTE@

Extracts the minute value from a serial time number

Format MINUTE@(timeNumber)

Arguments timeNumber A time serial value.

Description MINUTE@ is a time function which extracts the minute (0-59) value from a serial time number. You can enter a serial time number as an argument for the MINUTE@ function. A formula that contains the serial time number 0.129270833 (which represents a time of 3 hours, 6 minutes, and 9 seconds) returns 6.

You can also use the NOW@ or TIME@ functions as arguments in the MINUTE@ function. For example, the formula MINUTE@(TIME@(20,21,22)) returns 21.

MONTH@

Extracts the month of the year from a serial date number

Format MONTH@(dateNumber)

Arguments dateNumber A date serial value.

Description MONTH@ is a date function which extracts the month of the year (1-12) from a serial date number. You can enter a serial date number as an argument for the MONTH@ function. A formula that contains the serial number 30899 (August 6, 1984) returns 8.

You can also use the **DATE@** or **TODAY@** function as an argument in the MONTH@ function. For example, the formula MONTH@(DATE@(84,7,1)) returns 7

MONO_TEMPLATE_DIR@

Returns the directory containing black and white Applixware Presents templates

Format MONO_TEMPLATE_DIR@()

Description Returns the directory containing templates used by Applixware Presents for black and white presentations on paper or transparencies.

MOVE_FILE@

Moves a file to a different location

Format MOVE_FILE@(name, destination)

Arguments

name	The name of the file to move.
destination	The path name of the directory to where you want to move the file.

Description Moves a file to a new location by copying it to the new location and deleting it from the present location.

Example

See also **COPY_FILE@**

NETWORKDAYS@

Calculates the number of work days between two dates

Format NETWORKDAYS@(startDate, endDate, [, Holiday1] [, Holiday2]...[,Holidayn])

Arguments

startDate	a number containing the date value of the starting date
EndDate	a number containing the date value of the ending date
Holidayn	The date value of a holiday to be excluded from the total number of days calculated.

Description NETWORKDAYS@ calculates the number of whole working days between startDate and endDate. Working days exclude weekends (Saturday and Sunday) and any days identified as holidays.

The startDate and endDate arguments are numbers representing the number of days after 12/31/1899 that the given date falls. You can get the value for a date by passing the formatted date string to the DATEVALUE@ macro. For example, DATEVALUE@("1/1/00") is 1, and DATEVALUE@("5/18/60") is 22054.

For holidays, you may enter an optional set of one or more date values representing days to exclude from the working calendar. The following macro returns the number of workdays between May 1st and July 31st excluding Memorial Day and the July 4th holiday.

```
macro HowManyDays
var days
days = NETWORKDAYS@(DATEVALUE@("05/01/95"), DATEVALUE@("07/31/95"),
DATEVALUE@("05/31/95"), DATEVALUE@("07/04/95"))
info_message@("the number of days is "++days)   ' returns 64
endmacro
```

See also DATEVALUE@

NEW_TASK@

Starts a new ELF task

Format id =NEW_TASK@(macro[, arg1,...,argN)

Arguments macro — The macro name that runs the task you want to perform.

args — An optional series of arguments that is passed to macro. The elements of args can consist of strings or numbers, but either way they are passed to the calling macro as strings.

Numbers are converted to strings when they are passed to macro.

Description NEW_TASK@ starts an ELF task. This task runs until it gives up control of the thread. The new task can give up control of the thread by calling one of two macros: DELAY@() and DB_DISPLAY@().

NEW_TASK@ is similar to the macro PEND_FOR_NEW_TASK@. The only difference between NEW_TASK@ and PEND_FOR_NEW_TASK@ is that tasks spawned with NEW_TASK@ can give up control of the execution thread, where tasks spawned with PEND_FOR_NEW_TASK@ must complete in order to give up control of the thread.

The example macros illustrate the use of NEW_TASK@. Each time the DELAY@ statement is encountered, control of the execution thread passes from one macro to the other.

NEW_TASK@ returns the task id for the newly created task.

See also PEND_FOR_NEW_TASK@

Example

NEW_TASK_UNPENDED@

Starts a new ELF task

Format id =NEW_TASK_UNPENDED@(macro[, arg1,...,argN)

Arguments macro — The macro name that runs the task you want to perform.

args — An optional series of arguments that is passed to macro. The elements of args can consist of strings or numbers, but either way they are passed to the calling macro as strings.

Numbers are converted to strings when they are passed to macro.

Description NEW_TASK_UNPENDED@ starts adds an ELF task to the scheduling queue. The current task then continues execution with the next statement after NEW_TASK_UNPENDED@. The new task runs only after the current task yields control of the execution thread, and the ELF scheduler allocates some time to it.

For example, the following macro adds two tasks to the scheduling queue:

```
macro AddTwo
PRINTF@("One Task is running...")
     NEW_TASK_UNPENDED@("AddOne")
     NEW_TASK_UNPENDED@("AddAnother")
/*
*    There are now two tasks waiting on the scheduling
*    queue.  This task is still in control of the execution
*    thread.
*/
PRINTF@("This task is running, two tasks are waiting.")
endmacro
```

Tasks queued with NEW_TASK_UNPENDED@ are run by the ELF Scheduler in the order that they are queued. In the example, AddOne would run before AddAnother.

NEW_TASK_UNPENDED@ returns the task id for the newly scheduled task.

See also New_Task@, PEND_FOR_NEW_TASK@, ELF Scheduler

NOW@

Returns a decimal value that corresponds to the current date and time

Format NOW@( )

Description NOW@ is a time function which returns the decimal value that corresponds to the current date and time. The current date is indicated by the numbers to the left of the decimal point and the current time is represented by the numbers to the right of the decimal point. For example, entering the formula NOW@() on April 9, 1992 at a time of 13 hours, 25 minutes, and 48 seconds returns 33702.559583333.

This function does not require an argument; however, it must be followed by open and closed parentheses. If you are using this function in a Spreadsheet, you should have the cell's style set to a Date style.

If you want this function to update periodically in a Real Time Spreadsheet you can use this function with the ONTIME Spreadsheet function. For example, to display the current time every 15 seconds use the following:

=IF(ONTIME(-1,15), NOW@(), 0)

See also ONTIME and NOW Spreadsheet functions.

NUMBERSTYLE@

Indicates numbering style

Format flag = NUMBERSTYLE@()

Description Returns the user's current **Decimal Separator Character** preferences setting. This setting controls the behavior of periods and commas when displaying numbers. It returns FALSE if the Decimal Separator Character preference in a user's ax_prof4 file is set to display numbers in American Style. In the style, commas separate thousands and the period separates whole numbers from decimals; for example, 2,000,000.00.

It returns TRUE if this is set to display numbers in "European style". In this style, the period separates thousands and the comma separates whole numbers from decimals; for example, 2.000.000,00.

See also The **Decimal Separator Character** and **Place the Currency Symbol Left of the Number** International preference options.

Example

NUMBERSTYLESTR@

Returns the European-style of an American-style numeric string

Format val = NUMBERSTYLESTR@(string)

Arguments	string	A numeric string in ``American'' format (where commas separate thousands and a period separates whole numbers from decimals).

Description Returns the European-style equivalent of an American-style numeric string. Commas are replaced with periods, and periods are replaced with commas. Your **Decimal Separator Character** (axNumberStyle) preference must be TRUE for NUMBERSTYLESTR@ to work properly. If axNumberStyle is FALSE, NUMBERSTYLESTR@ returns a string identical to string.

See also **NUMBERSTYLEVAL@**, and **Place the Currency Symbol Left of the Number** International preference option

NUMBERSTYLEVAL@

Returns the decimal value of a European-style numeric string

Format val = NUMBERSTYLEVAL@(string)

Arguments	string	A numeric string in ``European'' format (where periods separate thousands and a comma separates whole numbers from decimals).

Description Converts a numeric string expressed in European format to its decimal value. All periods are deleted, and the comma is converted to a period. Your axNumberStyle preference must be TRUE for NUMBERSTYLEVAL@ to work properly. If axNumberStyle is FALSE, NUMBERSTYLEVAL@ returns a string identical to string.

See also NUMBERSTYLESTR@. Decimal Separator Character and Place the Currency Symbol Left of the Number International preference option

NUM_TO_STRING@

Returns a string representing a number's ASCII character equivalent

Format string = NUM_TO_STRING@(number)

Arguments	number	A string consisting of a decimal number or hexadecimal number. If number is hexadecimal, the hexadecimal number must be preceded by ``0x'' or ``0X'' (a zero followed by an ``x'').

Description Returns a string corresponding to the ASCII character equivalent of the number or hexadecimal number you specify.

Example

See also STRING_TO_NUM@
WP_ENTER_CHAR@

OPEN_ASCII_FILE@

Opens an operating system file

Format OPEN_ASCII_FILE@(filename, mode)

Arguments	
filename	The full path name of the operating system file you want to open.
mode	A string representing the access mode of the file. Specify ``r" to read the file, ``w" to create the file, and ``a" to append the data to the end of the file.

Description Opens an operating system file. If the specified file does not exist, and you open it in ``w" or ``a", a new file is created. If the specified file does not exist, and you open it with an ``r", an error occurs.

An operating system file opened using OPEN_ASCII_FILE@ must be closed using CLOSE_FILE@. If a macro that opens and closes an operating system file terminates before the file can be closed, the file remains open. If you attempt to reopen an operating system file that has not been closed, ELF throws an error.

If the file is opened in write mode (mode = w) or append mode (mode = a), a write lock is placed on the file. Similarly, if you open the file in read mode (mode = r), a read lock is placed on the file. File locks are cleared when you close the file using the CLOSE_FILE@ macro.

Because no more than 10 files can be open at any one time, rerunning a macro that does not close a file will eventually produce a condition where an error is thrown indicating that too many files are open.

Example

See also CLOSE_FILE@
READ_FILE@
WRITE_FILE@

OPEN_DOC@

Opens a new or existing Applix*ware* document

Format OPEN_DOC@(filename [, menuId][, windowlessFlag] [, read_only] [, urlHost] [,urlPort] [,username] [,password])

Arguments filename — The target document. If this argument is a pathname, it is treated as a file on the local system. If this argument is a URL, such as http://www.applix.com, it is treated as a file on the internet.

The type of window opened depends on the file type. Many different files can be opened with Open_Doc@. A default list of files can be found in the file hooks.dat. You can extend this list by adding your own entries to hooks.dat.

menuId — A unique number identifying the application window's menu bar as set using SET_SELECTIONS@. Set to NULL to use the default menu bar. Possible ids are:

100-199	opens a Spreadsheets window
200-299	opens a Words window
300-399	opens a Graphics window
400-499	opens a Macros window

windowlessFlag
Indicates whether the application window should display its window or not. Specify TRUE to run the application windowless. Specify FALSE (the default) to run the application with its window.

readOnlyFlag — A Boolean value which is set to TRUE to indicate that the document should be brought up on a *read-only* state (if the document type has a *read-only* state). If a document is read-only, you cannot overwrite the original document, and the File Ý Save option is grayed out.

urlHost — The name or IP address of the HTTPD host or proxy server through which you access the internet. If this field is a name, that name must be present in the file etc/hosts on your local machine. This is an optional parameter, and is only used when the file name is a URL.

urlPort — The port corresponding to the urlHost. This is an optional parameter, and is only used when the file name is a URL.

username — This name is used when URL authentication is requested by the target server for the file you are trying to access. This is an optional parameter, and is only used when the file name is a URL.

password — This password is used when URL authentication is requested by the target server. This is an optional parameter, and is only used when the file name is a URL.

Description OPEN_DOC@ opens a document in Applix*ware*. The document can be either a file on the local file system, or a file on the internet.

Opening Local Files

If the filename parameter is the name of a file on the local file system, OPEN_DOC@ opens a new Applix*ware* document window. The application opened depends on the file type provided. The newly opened window becomes the active window.

If filename does not specify the entire path name, the path defaults to the current directory. For best results, specify the entire path name.

menuId is useful for displaying two application windows whose menu bars are visibly different even though the windows bear the same application. To display such a window, first load the menu bar into memory using SET_SELECTIONS@.

The windowlessFlag option lets you perform automated tasks ``in the background,'' without displaying the application window at all. The windowless application becomes a ``child'' of the window from which it was invoked. With windowlessFlag, two applications can work simultaneously without interrupting each other, and without invoking two separate axmain processes. You also save time by not displaying and constantly refreshing the windowless application.

To perform a windowless task:

· You must suppress info and error messages immediately upon invoking the windowless application. Otherwise, the task hangs' when it attempts to display such a message. To suppress *info* and error messages, use SUPPRESS_INFO_MESSAGES@ and SUPPRESS_ERROR_MESSAGES@.

· You cannot include any (PROMPT@) prompts.

· Be sure to explicitly exit any windowless application after it completes its task. Otherwise, the task remains in memory until you log out.

· It is best to make a call to SELECT_WINDOW@ when exiting the windowless application and returning to the parent window.

· Before testing a new macro that invokes a windowless application, first test the macro with all windows displaying in the foreground. Windowless applications can also be created using the -quiet and -add command-line options.

Opening a File Across the Internet

If the filename parameter is a URL, such as http://www.applix.com, OPEN_DOC@ downloads the file referenced by the URL from the internet, performs any necessary conversions on the file, and opens it.

If no urlhost and urlport are specified in the OPEN_DOC@ command line, OPEN_DOC@ uses the server name and port you specified in the URL Preferences dialog box to initiate the download. It converts the file with the filter macro specified for the given file type in hooks.dat. For example, the default entry for HTML uses the filter macro HTML_OPEN_HTML@.

```
35:html:(html_open_html@ <#n#>)
```

OPEN_DOC@ tries to use an appropriate application for opening the file. For example, if the file is an HTML file, OPEN_DOC@ opens the file using the HTML Author; if the file is a GIF or JPEG file, OPEN_DOC@ opens the file using Applixware Graphics, and so on.

Hooks.dat

Using hooks.dat, you can specify the application with which to open a particular file type. For example, if you want to open JPEG files downloaded from the internet with an application other than Applix*ware*, you can configure hooks.dat to do this as follows:

84:jpeg:/bin/xv <#n#>

Hooks.dat is located by default in *install_dir*/axdata, where *install_dir* is the name of the directory where Applix*ware* is installed. You can copy hooks.dat to *install_dir*/axlocal or to your Applix*ware* home directory to edit the file.

There are three fields of information information for each file type in hooks.dat:

file type:bitmap:command

File type is specified by the file extension or by the Applix*ware* magic number defined in axmagic (located in axdata) and returned by the Applixware file recognition software.

Bitmap is the name of a 16X16 Applixware bitmap which is displayed as a graphical representation of the attached file of this type.

command is the command to be executed to open this file type. This field can contain a UNIX command or the name of an ELF macro to be called when files of this type are opened. By default the file is imported into the appropriate Applix*ware* application. However, you can run third-party software by setting up the appropriate command for those applications.

For more information on hooks.dat, refer to "Importing and Launching Non-Applix*ware* files" in Chapter 7 of the Applixware System Administrator's Guide.

See also GR_APPLICATION_DLG@

ME_APPLICATION_DLG@

SS_APPLICATION_DLG@

WP_APPLICATION_DLG@

SET_SELECTIONS@

SUPPRESS_INFO_MESSAGES@

SUPPRESS_ERROR_MESSAGES@

Command Line Options

OPEN_PROMPT@

Displays an Open dialog box and returns the pathname of a file

Format OPEN_PROMPT@(suffix, dir, wildcard, title, helpID)

Arguments

suffix	A string containing the file name suffix (extension) which is displayed in the Search entry box in the Open dialog box.
dir	The directory name whose contents are displayed in the Open dialog box list box. If dir is not included, the current directory's files are displayed.
wildcard	The file matching criteria, to be displayed in the Search entry box when the Open dialog box is first displayed. Only those files matching wildcard are displayed.
title	The title to be used for the Open dialog box.
helpID	The name assigned to the help text for this dialog box. This name is treated the same as any other dialog box name within the Help system.

Description Displays the Open dialog box with its list box containing files that meet the attributes you specify using dir and wildcard. After the user makes a selection, OPEN_-PROMPT@ returns the pathname of the selected file selected. Open dialog box includes a list box, file matching criteria entry box, a Directory option button, a Jump button, and Open, Cancel, Search, and Help push buttons.

OUTLINE_TEMPLATE_DIR@

Returns the directory containing
outline text for Applixware Presents

Format OUTLINE_TEMPLATE_DIR@()

Description Returns the directory containing the ooutline text for Applixware Presents. Outline text is used along with template backgrounds to create a structure for your presentation. The default outline text directory is /*install_dir*/template/outline/*lang*, where *install_dir* is your Applix*ware* install directory, and *lang* is your language directory.

PARSE_PATHNAME@

Returns directory and file portions of path name

Format PARSE_PATHNAME@(path, dir, file)

Arguments

path	Path name to break into directory and file components.
dir	Directory portion of path; returned by PARSE_PATHNAME@.
file	File portion of path; returned by PARSE_PATHNAME@.

Description The directory and file components are returned into the dir and file arguments to PARSE_PATHNAME@.

Example

See also ABSOLUTE_PATH@
CURRENT_DIR@
SPLIT_PATHNAME@

PEND_FOR_NEW_TASK@

Starts and completes an Applix*ware* task

Format [data=] PEND_FOR_NEW_TASK@ (macro[, arg1,...,argN])

Arguments

macro	The name of the macro that runs as a child task.
args	A series of arguments that is passed to macro. The elements of args can consist of strings or numbers, but are passed to the calling macro as strings. For example, if args[0] is 57, it is converted to ``57'' when passed to the macro. Elements in the args array cannot contain any embedded spaces (or else each word space breaks the element into two elements).

Description Runs a specified macro as a new ELF task. The macro that calls PEND_FOR_NEW_TASK@ is completely blocked until the child task completes. The child task cannot give up the execution thread.

The task created by this macro can return information to the calling macro using the ELF Return statement.

Example

See also NEW_TASK@, TASKS

PI@

Returns the value of pi

Format PI@( )

Description Return the value of pi.

PICK_EXTEND_SELECT@

Extends an existing selection

Format PICK_EXTEND_SELECT@(xpos, ypos[,numQuick])

Arguments

xpos	A number indicating the x-axis position of the mouse pointer in the work area. Position is in pixels and is relative to the top left corner of the work area.
ypos	A number indicating the y-axis position of the mouse pointer in the work area. Position is in pixels and is relative to the top left corner of the work area.
numQuick	If the mouse click represents a quick click, this number indicates the number of quick clicks after the first.

Description Extends an existing selection in the work area using a mouse click.

See also PICK_MULTI_SELECT@
PICK_PASTE@
PICK_SELECT@

PICK_MULTI_SELECT@

Creates multiple selections

Format PICK_MULTI_SELECT@(xpos, ypos[,numQuick])

Arguments

xpos	A number indicating the x-axis position of the mouse pointer in the work area. Position is in pixels and is relative to the top left corner of the work area.
ypos	A number indicating the y-axis position of the mouse pointer in the work area. Position is in pixels and is relative to the top left corner of the work area.
numQuick	If the mouse click represents a quick click, this number indicates the number of quick clicks after the first.

Description Creates multiple discontiguous selections in the work area using mouse clicks.

See also PICK_EXTEND_SELECT@
PICK_PASTE@
PICK_SELECT@

PICK_PASTE@

Pastes information

Format PICK_PASTE@(xpos, ypos)

Arguments

xpos	A number indicating the x-axis position of the mouse pointer in the work area. Position is in pixels and is relative to the top left corner of the work area.
ypos	A number indicating the y-axis position of the mouse pointer in the work area. Position is in pixels and is relative to the top left corner of the work area.

Description Pastes information at a location also sending a mouse pick.

See also PICK_EXTEND_SELECT@
PICK_MULTI_SELECT@
PICK_SELECT@

PICK_SELECT@

Performs a mouse click in an application's work area

Format PICK_SELECT@(xpos, ypos[,numQuick])

Arguments

xpos	A number indicating the x-axis position of the mouse pointer in the work area. Position is in pixels and is relative to the top left corner of the work area.
ypos	A number indicating the y-axis position of the mouse pointer in the work area. Position is in pixels and is relative to the top left corner of the work area.
numQuick	If the mouse click represents a quick click, this should be a number indicating the number of quick clicks after the first.

Description Used, for example, to position the cursor in a window. The coordinates are measured in pixels. The resolution of most screens is approximately 75 dots per inch (DPI). Therefore, PICK_SELECT@(75, 75) would place the mouse approximately one inch down and one inch to the right of the top left corner.

See also PICK_EXTEND_SELECT@

PICK_MULTI_SELECT@

PICK_PASTE@

POWER@

Raises a positive number to a power or a negative number
to an integer power

Format POWER@(value, power)

Arguments

value	The value to be raised.
power	A number indicating the desired power.

Description Raises a positive number to a positive or negative power or a negative number to an integer power. For example, POWER@(3,2) raises the value 3 to the power 2. It returns the value 9.

See also EXP@

PREFERENCES@

Returns an Applix*ware* preference variable value

Format value = PREFERENCES@(prefVar)

Arguments	prefVar	A string, the preference variable name for which you want the value returned. The file ax_prof4 lists all preference editor variables and their values in the format prefVar : value.

Description Returns a string indicating the value of the profile variable specified. PREFERENCES@ returns NULL if the specified variable does not have a value.

Example

See also CHANGE_PREFS@

EDITPREFS@

PRESENTS_DLG@

Brings up the Presentation Editor Startup Dialog

Format PRESENTS_DLG@()

Description Brings up the Applixware Presents Startup dialog. This dialog allows you to specify startup parameters for Applixware Presents.

print_baggage@ format

Description The print_baggage@ format is used to establish printer configuration information for a print job. Many of the strings used in the print_baggage@ format are defined in the **ax-pdf** file. This file is located in the install_dir/axlocal directory, and is described in the System Administration Guide.

Print_baggage@ is defined in the ELF include file print_.am. The fields in the print_baggage@ format are as follows:

format PRINT_BAGGAGE

PRINT_TO_FILE,	-1 to redirect the print job to a file. 0 to send the print job to a printer.
REVERSE,	-1 to print the file from last page to first page. 0 to print the file from first page to last page.
DUPLEX,	An integer that indicates the duplex setting for the print job. -1 means use the document settings 0 means simplex 1 means duplex, long-side binding 2 means duplex, short-side binding
SOURCE_ALL,	A Postscript/PCL string indicating the source paper tray for the print job. This string must be one of the strings in the SOURCE array. If this string is set to NULL, the document setting is used.
SOURCE_1ST,	A Postscript/PCL string indicating the source paper tray for the first page of the print job. This string must be one of the strings in the SOURCE array. If this string is set to NULL, the SOURCE_ALL value is used.
SOURCE_LAST,	A Postscript/PCL string indicating the source paper tray for the for the last page of the print job. This string must be one of the strings in the SOURCE array. If this string is set to NULL, the SOURCE_ALL value is used.
DEST_ALL,	A Postscript/PCL string indicating the output bin for the print job. This string must be one of the strings in the DEST array. If this string is set to NULL, the document setting is used.
DEST_1ST,	A Postscript/PCL string indicating the output bin for the first page of the print job. This string must be one of the strings in the DEST array. If this string is set to NULL, the DEST_ALL setting is used.
DEST_LAST,	A Postscript/PCL string indicating the output bin for the last page of the print job. This string must be one of the strings in the DEST array. If this string is set to NULL, the DEST_ALL setting is used.
SOURCE,	An array containing all the possible sources for the printer. These codes are listed in the PaperSource section of the axpdf file. For example:

```
PaperSource:
  Auto        [PCL: "<esc>&l7H"]
  Upper Drawer [PCL: "<esc>&l1H"] [PS: "*InputSlot Tray2/Tray 2"]
  Lower Drawer [PCL: "<esc>&l4H"] [PS: "*InputSlot Tray3/Tray 3"]
  Manual      [PCL: "<esc>&l2H"] [PS: "*ManualFeed True/True"]
  Manual Envelope [PCL: "<esc>&l3H"]
  Envelope Feeder [PCL: "<esc>&l6H"] [PS: "*InputSlot Envelope/Envelope Feeder"]
  Right Tray   [PCL: "<esc>&l8H"] [PS: "*InputSlot Tray1/Tray 1"]
  HCI Tray 1   [PCL: "<esc>&l5H"]
  HCI Tray 2   [PCL: "<esc>&l20H"]
```

If the target printer is a PCL printer, the SOURCE array should contain the values in quotes in each of the PCL: fields. If the target printer is a postscipt printer, the SOURCE array should contain the values in quotes in each of the PS: fields.

DEST, An array containing all the possible paper tray destinations for the printer. These codes are listed in the PaperDestination section of the axpdf file. For example:

```
PaperDestination:
  Auto          [PCL: "<esc>&l0G"]
  Face Down          [PCL: "<esc>&l1G"] [PS: "*OutputBin Upper/Top Output Bin (Face Down)"]
  Face Up            [PCL: "<esc>&l2G"] [PS: "*OutputBin Left/Left Output Bin (Face Up)"]
  HCO Face Up        [PCL: "<esc>&l2G"]
  HCO Bin 1          [PCL: "<esc>&l4G"]
```

If the target printer is a PCL printer, the DEST array should contain the values in quotes in each of the PCL: fields. If the target printer is a postscipt printer, the DEST array should contain the values in quotes in each of the PS: fields.

SIZE, An array containing all the possible paper sizes for the printer. These codes are listed in the Papersize section axpdf file. For example:

```
PaperSize:
  Letter    (8.5"  11")          [PS: "*PageSize Letter/US Letter"]
  Executive (7.25" 10.5")    [PS: "*PageSize Executive/Executive"]
  Legal     (8.5"  14")                [PS: "*PageSize Legal/US Legal"]
  Tabloid   (11"   17")                [PS: "*PageSize Tabloid/11x17"]
  Tabloid.2 (11"   17")                [PS: "*PageSize Tabloid.2/11x17 (Oversize)"]
  A4        (210mm 297mm)              [PS: "*PageSize A4/A4"]
  A3        (297mm 420mm)              [PS: "*PageSize A3/A3"]
  B4     (257mm 364mm)   [PS: "*PageSize B4/JIS B4"]
  International B5 (182mm  257mm)       [PS: "*PageSize B5/JIS B5"]
  Postcard    (5.833" 7.875")           [PS: "*PageSize JDPost/Double Postcard"]
  Com-10 Business (4.125" 9.5")  [PS: "*PageSize Comm10/Env Comm10"]
  Monarch         (3.875" 7.5")         [PS: "*PageSize Monarch/Env Monarch"]
  International DL (110mm  220mm)       [PS: "*PageSize DL/Env DL"]
  International C5 (162mm  229mm)       [PS: "*PageSize C5/Env C5"]
  International B5 (176mm  250mm)       [PS: "*PageSize EnvB5/Env ISO B5"]
```

The SIZE array should contain the values in quotes in each of the PS: fields. The Papersize settings do not apply to PCL printers.

PPDFILE,	A string containing the Postscript Printer Definition file name. This string is listed in the axpdf file.
DUPLEX_CODES	An array containing all the possible Duplex options for the printer. These codes are listed in the Duplex section of the axpdf file. For example:

```
Duplex: Yes
  Simplex             [PS: "*Duplex None/Off"]
  Duplex Long-Edge         [PS: "*Duplex DuplexNoTumble/Long-Edge Binding"]
  Duplex Short-Edge        [PS: "*Duplex DuplexTumble/Short-Edge Binding"]
```

If the target printer is a postscipt printer, the DUPLEX_CODES array should contain the values in quotes in each of the PS: fields. The Duplex settings do not apply to PCL printers.

See also The discussion of the **axpdf** file in the *System Administration Guide*.

PRINTF@

Prints arguments in a window

Format PRINTF@(a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p)

Arguments a..p Any ELF string or number (or expression or macro that evaluates to a string or number).

Description Prints information to the terminal window from which you invoked Applix*ware*. Because this function writes information out to the window as a stream, you may want to append "\n" to the string being sent so that it is separated into lines.

For example:

foo = "dog"

foo1 = "cat"
printf@("Quick brown %s and %s\n", foo, foo1)

This prints "Quick brown dog and cat".

As this example illustrates, the string being printed contains one or more formatting code/variable pairs embedded within a string. In many cases, the text to be printed contains information other than the formatting code. However, you can omit this additional information.

For information on using formatting codes, see **FORMAT@**.

PRINT_FILE@

Prints a file

Format PRINT_FILE@(filename, printer, colorFlag, copies, bannerFlag, allPagesFlag, startPage, endPage, NULL, viewName, mergeFile, tempFile, NULL, NULL, NULL, startText, endText, class, deleteWhenDoneFlag, baggage)

Arguments

filename	The name of the file being printed.
printer	A string giving the printer name to print the document. If you want to print to a file rather than to a printer, set printer to NULL; the print file is placed in the directory specified by tempFile, or in the Applix*ware* temporary directory if no directory is specified in tempFile.
colorFlag	Indicates whether the document is being printed on a color printer. If you are printing to a color printer, set colorFlag to TRUE; otherwise, set colorFlag to FALSE, which is the default.
copies	The number of copies to print. The default is 1.
bannerFlag	Indicates whether to include a banner page with the printed document. Set to TRUE if you want a banner page; FALSE if you don't, (which is the default).
allPagesFlag	Indicates whether to print all pages of the document. Set to TRUE to print all pages; FALSE if you want to print a range of pages.
startPage	A number indicating the first page to print if you are printing a range of pages. If allPagesFlag is TRUE, startPage is ignored.
endPage	A number indicating the last page to print if you are printing a range of pages. If allPagesFlag is TRUE, endPage is ignored.
viewName	The view name within a spreadsheet.
mergeFile	The name of the file to merge with the document when printing. Specify NULL if you are not using a merge file.
tempFile	The name of the print file. This file is created to print the document and is deleted when printing is completed (unless printer is set to NULL). If you do not supply a path name for temp_file, the print file is created in the Applix*ware* temporary directory.
startText	The beginning page of any *lateral* pages to be printed. For example, if this document includes a wide spreadsheet inset that spans from page 1a to 2b, type 1a. Set to NULL if no lateral pages are includes.

endText	The ending page of any *lateral* pages to be printed. For example, if this document includes a wide spreadsheet inset that spans from page 1a to 2b, type 2b. Set to NULL if no lateral pages are includes.
class	A constant indicating if printing will be performed on a PostScript or PCL5 device, as follows: PostScript PCL5
deleteWhenDoneFlag	A Boolean value which if set to TRUE indicates that filename will be deleted after it is printed.
baggage	an array of format **print_baggage@**.

Description Prints a file, after being passed a file name and other information.

Example

See also **PRINT_FILE_BACKGROUND@**

PRINT_FILE_BACKGROUND@

Prints an Applix*ware* document after starting a new server process

Format PRINT_FILE_BACKGROUND@(filename, printer, colorFlag, copies, bannerFlag, allPagesFlag, startPage, endPage, NULL, viewName, mergeFile, tempFile, NULL, NULL, NULL, startText, endText, class, deleteWhenDoneFlag, baggage)

Arguments

filename	The name of the file being printed.
printer	A string giving the printer name to print the document. If you want to print to a file rather than to a printer, set printer to NULL; the print file is placed in the directory specified by tempFile, or in the Applix*ware* temporary directory if no directory is specified in tempFile.
colorFlag	Indicates whether the document is being printed on a color printer. If you are printing to a color printer, set colorFlag to TRUE; otherwise, set colorFlag to FALSE, which is the default.
copies	The number of copies to print. The default is 1.
bannerFlag	Indicates whether to include a banner page with the printed document. Set to TRUE if you want a banner page; FALSE if you don't, (which is the default).

allPagesFlag	Indicates whether to print all pages of the document. Set to TRUE to print all pages; FALSE if you want to print a range of pages.
startPage	A number indicating the first page to print if you are printing a range of pages. If allPages is TRUE, startPage is ignored.
endPage	A number indicating the last page to print if you are printing a range of pages. If allPages is TRUE, endPage is ignored.
viewName	The view name within a spreadsheet.
mergeFile	The name of the file to merge with the document when printing. Specify NULL if you are not using a merge file.
tempFile	The name of the print file. This file is created to print the document and is deleted when printing is completed (unless printer is set to NULL). If you do not supply a path name for temp_file, the print file is created in the Applix*ware* temporary directory.
startText	The beginning page of any *lateral* pages to be printed. For example, if this document includes a wide spreadsheet inset that spans from page 1a to 2b, type 1a. Set to NULL if no lateral pages are includes.
endText	The ending page of any *lateral* pages to be printed. For example, if this document includes a wide spreadsheet inset that spans from page 1a to 2b, type 2b. Set to NULL if no lateral pages are includes.
class	A constant indicating if printing will be performed on a PostScript or PCL device, as follows: 0 PostScript 1 PCL
deleteWhenDoneFlag	A Boolean value which if set to TRUE indicates that filename will be deleted after it is printed.
baggage	an array of format print_baggage@.

Description Prints an Applix*ware* document without interrupting other Applix*ware* tasks. This macro starts another axmain server process for the print job. The new axmain does not check out an additional license and it terminates when it completes printing the file. This macro is recommended for particularly long print jobs. Avoid using it to print short documents: it may take more time to start the new axmain process than to print in the foreground.

Example

See also PRINT_FILE@

PROMOTE_APP_WINDOW@

Raise an application to the top

Format flag = PROMOTE_APP_WINDOW@(name[, parentTask[, noPromoteFlag]])

Arguments

name	The application window to promote.
parentTask	The task that spawned the application window. If you include this argument, the window is only promoted if it was created by this parentTask.
noPromoteFlag	A Boolean value which if set to TRUE indicates that the window should not be promoted. In this case, you are using this macro to determine if the window exists.

Description Ensures that an application window is visible by displaying it on top of all other windows. Its *X* and *Y* position on the screen is not changed.

If the window specified by name exists, PROMOTE_DIALOG@ places it in front of all others on the screen. If the window does not exist, PROMOTE_DIALOG@ returns FALSE.

If you include the optional parentTask argument and this task does not exist, FALSE is also returned.

See also PROMOTE_DIALOG@

PROMOTE_DIALOG@

Raises a dialog box to the front of the screen

Format flag =PROMOTE_DIALOG@(name[, parentTask[, noPromoteFlag]])

Arguments

name	The dialog box title to promote.
parentTask	The task that spawned the dialog box. If you include this argument, the dialog box is only promoted if it was created by this parentTask.
noPromoteFlag	A Boolean value which if set to TRUE indicates that the dialog box should not be promoted. In this case, you are using this macro to determine if the window exists.

Description Ensures that a dialog box is visible by displaying it on top of all other windows. Its *X* and *Y* position on the screen is not changed.

If the dialog box window specified by name exists, PROMOTE_DIALOG@ places the dialog box in front of all others on the screen. If the specified dialog box does not exist, PROMOTE_DIALOG@ returns FALSE.

If you include the optional parentTask argument and this task does not exist, FALSE is also returned.

See also PROMOTE_WINDOW@

Example

PROMOTE_WINDOW@

Uncovers an application window completely visible

Format PROMOTE_WINDOW@(id)

Arguments		
	id	The task id of the window being raised.

Description Raises the application window so that is completely visible. That is, it is moved on top of any other application windows.

Only windows associated with Applix*ware* windows can be raised. Those associated with dialog box (such as Applixware Data and Applixware Mail) can not be raised. For these windows, use PROMOTE_DIALOG@.

PROMPT@

Returns the contents of an entry box

Format PROMPT@(promptText[,override[, defaultText[, maxLength]]])

Arguments		
	promptText	A string value that is displayed as the entry box label in the dialog box. This text is normally used to prompt the user for information which will be typed in the entry box.
	override	An optional variable that allows you to suppress the display of the prompt dialog box. If override is NULL, the prompt dialog box is displayed. If override is not NULL, the prompt dialog box is not displayed and the value of override is used as the return value for PROMPT@. This option is useful in instances when you only want a prompt dialog box to appear if the value for an argument has not been determined through other means.

defaultText	If desired, text can be displayed in the entry box when the dialog box is first displayed. defaultText is a string indicating the text that should be displayed in the entry box. If no change is made to default_text, and the OK button is pressed, defaultText, will be returned.
maxLength	A number indicating the maximum number of characters that may be typed in the entry box. If maxLength is greater than the number of characters that can be displayed in the entry box, the entry box will scroll horizontally as the characters are typed.

Description Displays a dialog box containing an entry box and OK and Cancel buttons. PROMPT@ returns a string indicating the value of the entry box when OK is pressed.

Example

PR_GET_PAPER_CODES@

Returns an array of paper codes for a printer

Format pdfArray = PR_GET_PAPER_CODES@(printerName, printerClass)

Arguments

printerName	A string name of a printer as it appears in the Print dialog box.
printerClass	A string indicating the type of printer:

ps	postscript printer
pcl5	PCL printer

Description Returns an array containing printer attributes as defined in the installDir/axlocal/axpdf file. The returned array contains the following information:

pdfArray

- PaperSource
- PaperDestination
- PaperSize - a set of 4 element arrays. A descriptions if each element follows:
  - Paper Width (in mils, where 1 mil - 1/1000 of an inch)
  - Paper Length (in mils)
  - Duplex
  - Page Size

The fields in the axpdf file are described in **The Printer Definition File**," in Chapter 3 of the *System Administrator's Guide*.

RANDOM@

Returns a 32-bit random number using the UNIX *rand* command

Format num = RANDOM@()

See also **RANDOM_SEED@**

RANDOM_SEED@

Seeds the UNIX *rand* command with the time of day

Format RANDOM_SEED@()

See also **RANDOM@**

READ_ASCII_FILE@

Reads an ASCII file

Format stringArray = READ_ASCII_FILE@(filename)

Arguments	filename	The name (or path name) of the ASCII file you want to read.

Description Returns an array of strings containing data read from filename. If you need to read an entire file, READ_ASCII_FILE@ performs this task faster than **READ_FILE@**. If filename does not exist, ELF throws an error.

While the file is being read, READ_ASCII_FILE@ places a read lock on the file. This lock is cleared when the file read is complete. If two ELF macros try to read the same file simultanously, one of the ELF macros will throw an error indicating that the file is locked.

See also **WRITE_ASCII_FILE@**

READ_BINARY_FILE@

Reads a file and returns it as a binary object

Format fileStream = READ_BINARY_FILE@(filename)

Arguments filename — The name of the file to be read. filename can be any type of file.

Description Reads filename and returns its contents as a binary object. Although READ_BINARY_FILE@ is typically used for reading binary files created with WRITE_-BINARY_FILE@, it will read any file and convert it to a binary object.

While the file is being read, READ_BINARY_FILE@ places a read lock on the file. This lock is cleared when the file read is complete. If two ELF macros try to read the same file simultanously, one of the ELF macros will throw an error indicating that the file is locked.

See also WRITE_BINARY_FILE@

READ_DATA_FILE@

Reads an ELF data file into an ELF array

Format array = READ_DATA_FILE@(filename)

Arguments filename — The name (or path name) of the ELF data file. filename can be any file type, and is normally created using WRITE_DATA_FILE@ .

Description Opens filename, then reads and places data into an ELF array. When all information from the file is read, filename is closed.

While the file is being read, READ_DATA_FILE@ places a read lock on the file. This lock is cleared when the file read is complete. If two ELF macros try to read the same file simultanously, one of the ELF macros will throw an error indicating that the file is locked.

See also WRITE_DATA_FILE@
READ_ELF_FILE@

READ_ELF_FILE@

Reads an ELF file

Format format read_elf_format@ data = READ_ELF_FILE@(filename)

Arguments filename The name (or path name) of the ELF data file.

Description Opens filename, then reads and places data into an ELF array. When all information from the file is read, filename is closed. This macro is a replacement for READ_DATA_FILE@. READ_DATA_FILE@ will not operate correctly for ELF application data file such as those used by Applixware Data as they have a file header on them.

You can read an ELF data file that does not contain a header using this macro. The results will differ from READ_DATA_FILE@ in that the ELF data is now assigned to the data component of the read_elf_format@.

The definition of read_elf_format@ is as follows:

```
format read_elf_format@
    format write_data_format@ info,
    data                'content as an ELF data array

format write_data_format@
    comments,           'Array of strings to be added to file as comment
    grp_access,         '0-none, 1-read, 2-write
    all_access,         '0-none, 1-read, 2-write
                        'Specs for header line
    format afile_info file_header

format afile_info
    encoding,           'TRUE if file is encoded
    version,            'version number of current format
    docType,            'see recgfil_.am
    original_version, 'version number of original document
    minimum_version,        'last version capable of reading this file
    content_hint        'arbitrary hint string
```

If the file has a header and is not the current version, a filter is run to convert it to the current version.

While the file is being read, READ_ELF_FILE@ places a read lock on the file. This lock is cleared when the file read is complete. If two ELF macros try to read the same file simultanously, one of the ELF macros will throw an error indicating that the file is locked.

READ_FILE@

Reads an operating system data file one line at a time

Format line = READ_FILE@(filename)

Arguments

filename	The full path name of an operating system file to read.

Description Reads one line of filename which was previously opened using OPEN_ASCII_FILE@. Its return value is a string representing the line read. ELF throws the error ERR#CANNOT_READ_ after READ_FILE@ reads the last line in the file.

While the file is being read, READ_FILE@ places a read lock on the file. This lock is cleared when the file read is complete. If two ELF macros try to read the same file simultanously, one of the ELF macros will throw an error indicating that the file is locked.

Example

See also CLOSE_FILE@

OPEN_ASCII_FILE@

READ_GRAPHIC_BUFFER@

Reads graphic information and associates it with a handle

Format READ_GRAPHIC_BUFFER@(gfx, buffer)

Arguments

gfx	A graphics handle.
buffer	An internal variable containing the graphics information. This information was created using WRITE_GRAPHIC_BUFFER@.

Description Creates a graphic editor using the already allocated graphics handle gfx. The contents of this editor are buffer.

READ_GRAPHIC_FILE@

Reads a file onto the graphic

Format READ_GRAPHIC_FILE@(gfx, filename[, readOnly])

Arguments	gfx	A graphics handle.
	filename	The name of the file being read.
	readOnly	A Boolean value which if set to TRUE indicates that the contents of filename cannot be changed.

Description Reads the contents of a graphics file and assigns it to the already allocated graphics handle gfx.

While the file is being read, READ_GRAPHIC_FILE@ places a read lock on the file. This lock is cleared when the file read is complete. If two ELF macros try to read the same file simultanously, one of the ELF macros will throw an error indicating that the file is locked.

READ_PREFERENCES@

Reads the preferences file

Format READ_PREFERENCES@(filename)

Arguments filename The path name of the preferences file.

Description Reads the profile file. If the profile file was already read, rereading the file overwrites existing values.

While the file is being read, READ_PREFERENCES_FILE@ places a read lock on the file. This lock is cleared when the file read is complete. If two ELF macros try to read the same file simultanously, one of the ELF macros will throw an error indicating that the file is locked.

RECOGNIZE_FILE@

Returns the file type id of a file

Format id = RECOGNIZE_FILE@(filename)

Arguments filename The name of the file whose type is being determined.

Description Returns the file type id of filename. This file type is listed below. The definition of these constants can be found in recgfil_.am in your ELF include directory

AX_DOC_TYPE_ASCII	ASCII
AX_DOC_TYPE_DCA	DCA

AX_DOC_TYPE_APPLIX_MARKUP	Applixware Markup
AX_DOC_TYPE_INTERLEAF_ASCII	Interleaf (ASCII)
AX_DOC_TYPE_FRAME_MIF	Frame MIF
AX_DOC_TYPE_WP_51	Word Perfect 5.1
AX_DOC_TYPE_MS_WORD	MS Word
AX_DOC_TYPE_OFFICE_WRITER	OfficeWriter
AX_DOC_TYPE_WP_42	Word Perfect 4.2
AX_DOC_TYPE_SAMNA	Samna
AX_DOC_TYPE_VOLKS_WRITER	VolksWriter
AX_DOC_TYPE_LEADING_EDGE	Leading Edge
AX_DOC_TYPE_WORDSTAR	WordStar
AX_DOC_TYPE_WORDSTAR_2000	WordStar2000
AX_DOC_TYPE_XYWRITE	XYwrite
AX_DOC_TYPE_PFS_PROFESSIONAL	PfsProfessional
AX_DOC_TYPE_PFS_WRITE	PfsWrite
AX_DOC_TYPE_EXECUTIVE_WRITER	ExecutiveWriter
AX_DOC_TYPE_SPELL_BINDER	SpellBinder
AX_DOC_TYPE_PC_WRITE	PCwrite
AX_DOC_TYPE_Q_AND_A	QandA
AX_DOC_TYPE_MULTI_MATE	MultiMate
AX_DOC_TYPE_MANUSCRIPT	Manuscript
AX_DOC_TYPE_SUN_WRITE	SunWrite
AX_DOC_TYPE_ISLAND_WRITE	IslandWrite
AX_DOC_TYPE_DISPLAY_WRITE	DisplayWrite
AX_DOC_TYPE_RTF	Rich Text Format (RTF)
AX_DOC_TYPE_MS_WFW2	MS Word for Windows
AX_DOC_TYPE_ASTERX2	Applix3 --> aster*x2.1
AX_DOC_TYPE_WP_50	Word Perfect 5.0
AX_DOC_TYPE_WP_60	Word Perfect 6.0
AX_DOC_TYPE_XWD	XWD
AX_DOC_TYPE_FAX_M	FAX msb
AX_DOC_TYPE_EPSI	EPSI
AX_DOC_TYPE_HPGL	HPGL
AX_DOC_TYPE_X_BITMAP	XBitmap
AX_DOC_TYPE_PCX	PCX

AX_DOC_TYPE_SUN_RASTER	Sun raster
AX_DOC_TYPE_MAC_PAINT	MacPaint
AX_DOC_TYPE_MS_WINDOWS_BITMAP	MSWindows Bitmap
AX_DOC_TYPE_CGM	CGM
AX_DOC_TYPE_DXF	DXF
AX_DOC_TYPE_FAX_I	FAX lsb
AX_DOC_TYPE_IGES	IGES
AX_DOC_TYPE_TIFF_M	TIFF M
AX_DOC_TYPE_TIFF_I	TIFF I
AX_DOC_TYPE_GIF	GIF
AX_DOC_TYPE_PICT	PICT
AX_DOC_TYPE_TGA	TGA
AX_DOC_TYPE_ILBM	ILBM
AX_DOC_TYPE_GEM	GEM
AX_DOC_TYPE_PBM	PBM
AX_DOC_TYPE_PGM	PGM
AX_DOC_TYPE_PPM	PPM
AX_DOC_TYPE_PICT2	PICT2
AX_DOC_TYPE_WMF	Windows Metafile
AX_DOC_TYPE_WPG	Word Perfect Graphics
AX_DOC_TYPE_GP4	Group 4 fax
AX_DOC_TYPE_XPM	X pixmap
AX_DOC_TYPE_RAW	Raw Bitmap
AX_DOC_TYPE_IRIS	SGI Iris images
AX_DOC_TYPE_WKS	WKS/WK1
AX_DOC_TYPE_WK1	
AX_DOC_TYPE_SYLK	SYLK
AX_DOC_TYPE_DIF	DIF/XDIF
AX_DOC_TYPE_XDIF	
AX_DOC_TYPE_ASCII_GRID	ASCII grid
AX_DOC_TYPE_ASCII_COLUMN	ASCII into 1 column
AX_DOC_TYPE_ASCII_ROW	ASCII into 1 row
AX_DOC_TYPE_CSV	CSV
AX_DOC_TYPE_XLS3	XLS 3.0
AX_DOC_TYPE_XLS4	XLS 4.0
AX_DOC_TYPE_WK3	WK3
AX_DOC_TYPE_ASTERX_SS2	Applixware Spreadsheets 2.1
AX_DOC_TYPE_AUDIO	Audio
AX_DOC_TYPE_DATA	Data
AX_DOC_TYPE_COMPRESSED_AUDIO	

	Compressed Audio
AX_DOC_TYPE_DIRECTORY	Directory
AX_DOC_TYPE_CHAR_SPECIAL	Character special
AX_DOC_TYPE_BLOCK_SPECIAL	Block special
AX_DOC_TYPE_SYMBOLIC_LINK	Symbolic link
AX_DOC_TYPE_SOCKET	Socket
AX_DOC_TYPE_FIFO	Fifo
AX_DOC_TYPE_UNKNOWN	Unknown
AX_DOC_TYPE_EMPTY	Empty
AX_DOC_TYPE_COMPRESSED_DATA	Compressed data
AX_DOC_TYPE_BTOA_DATA	btoa'd data
AX_DOC_TYPE_MAC_HEADER	has Mac Finder header
AX_DOC_TYPE_ASTERX_WORDS	Applixware Words
AX_DOC_TYPE_ASTERX_GRAPHICS	Applixware Graphics
AX_DOC_TYPE_ASTERX_SS	Applixware Spreadsheets
AX_DOC_TYPE_ASTERX_MACROS	Applixware Macros
AX_DOC_TYPE_ASTERX_BINARY_SS	Binary Spreadsheets
AX_DOC_TYPE_ASTERX_BITMAP	Applixware Bitmap
AX_DOC_TYPE_ASTERX_EQUATIONS	Applixware Equations
AX_DOC_TYPE_ASTERX_ASCII	Applixware external ascii
AX_DOC_TYPE_ASTERX_BINARY	Applixware ext. binary
AX_DOC_TYPE_ASTERX_ELFARRAY	Applixware ELF array
AX_DOC_TYPE_ASTERX_QUERYDATA	Applixware Query

See also RECOGNIZE_FILE_INFO@

RECOGNIZE_FILE_INFO@

Returns information on a file

Format format recognize_info@ = RECOGNIZE_FILE_INFO@(filename)

Arguments filename A file in which you are searching for information.

Description Returns a recognize_info@ format that contains the following information:

```
format recognize_info@
    docType,      'The document type ID returned by
                  ' RECOGNIZE_FILE@
    appType,      'The Applixware id type, as follows:
                  0   None
                  1   Words
                  2   Graphics
                  3   Spreadsheets
                  4   Macro
                  5   Audio
                  6   Equation
                  7   Query
                  8   Bitmap
    filterMacro,  'The filter macro used to import a document into Applixware.
    launchMacro
                  'The macro launched when the user double-clicks on the document or
                  when opened.
```

See also RECOGNIZE_FILE@

RECORD_ON@

Indicates whether the keystroke recorder is on

Format flag = RECORD_ON@()

Description Returns TRUE if the keystroke recorder is on; FALSE if it is not on.

REGISTER_OBJECT@

Adds an object to the Applixware Registry

Format REGISTER_OBJECT@(object, name, overwrite_flag, source_pathname)

Arguments

object	an object reference.
name	the name by which the object can be addressed by other applications
overwrite_flag	If TRUE, the macro overwrites an existing registry entry with the same name. If FALSE, existing registry entries are not overwritten.
source_pathname	The absolute pathname of the Builder source file containing the object to be registered.

Description Adds an entry to the Applixware Registry. Typically, this macro is called from within the set initialize_event of a Builder object.

When you register an object with Applix*ware*, the register name of the object, as specified in the name argument, can be retrieved by other applications that call FIND_REGISTERED_OBJECT@.

See also FIND_REGISTERED_OBJECT@, UNREGISTER_OBJECT@

REGISTERED_OBJECT_APPLICATION@

Returns the name of the application containing the named object

Format REGISTERED_OBJECT_APPLICATION@(name)

Arguments

name	The registered name of an object

Description Returns the name of the application containing the named object. The object must have been previously registered with Applix*ware* using the function REGISTER_OBJECT@.

RELEASE_RESOURCE@

Unblocks a resource

Format RELEASE_RESOURCE@(name)

Arguments name — The name of a resource. This name is defined (and agreed to) by macros that use the resource.

Description Unblocks (releases) a resource that was acquired using GET_RESOURCE@. Because no time-out provision exists for acquired resources, you must use this macro to allow other tasks to reuse a blocked resource.

REPAINT_ALL@

Refreshes all Applix*ware* windows

Format REPAINT_ALL@([windowNum])

Arguments windowNum — Specifies which window will be refreshed. If this argument is not used, all windows are refreshed.

Description Refreshes all Applix*ware* window displays, including dialog boxes.

Example

See also REPAINT_WINDOW@

REPAINT_WINDOW@

Refreshes the current window

Format REPAINT_WINDOW@()

Description Repaints the current window. You would use REPAINT_WINDOW@ after performing an operation that distorts the screen display of a window.

See also REPAINT_ALL@

REPLACE_SUBSTR@

Replace strings within a string

Format newString = REPLACE_SUBSTR@(oldString, lookFor, subWith)

Arguments

oldString	The target string.
lookFor	The string that is being replaced.

	subWith	The string that is substituted for the lookFor string.

Description Replaces all occurrences of the string pattern lookfor with the string pattern subWith. For example:

REPLACE_SUBSTR@(``abcdabcd'',``abc'', ``cd'')

yields ``cddcdd''.

See also SUBSTRING@

RESOLVE_LINK_PATHS@

Changes link path information from relative to absolute

Format RESOLVE_LINK_PATHS@(filename, format doc_links_info@ info)

Arguments	filename	The name of the file containing links.
	info	A format containing all of a document's link information. (To obtain this list, use GET_LINKS_INFO@.)

Description Transforms the path names of link objects contained within a doc_links_info@ structure from relative to absolute.

REVISION_NUMBER@

Returns the major revision number

Format REVISION_NUMBER@( )

Description Returns the major revision number. For example, if this release is 4.1 (100), this macro would return 4.1.

ROUND@

Rounds a number to a specified number of decimal places

Format ROUND@(num, decimalPlaces)

Arguments	num	The number to be rounded.
	decimalPlaces	The number of decimal places to round num.

Description This function rounds a number num, to a specified number of decimal places. You can specify a maximum of 9 decimal places.

Values greater than or equal to 5 are rounded up. Values less than 5 are rounded down. If decimalPlaces is a positive value, num is rounded to the specified digit. For example, the ROUND@(23.456,2), equals 23.46. It rounds 23.456 to two decimal places.

If places is a negative value, n is rounded to the nth power of 10. -1 causes places to be rounded to the nearest multiple of 10^1. -2 causes n to be rounded to the nearest 10^2. For example:

ROUND@(23.567,-1) equals 20

ROUND@(57.8,-2) equals 100

ROUND@(1234,-3) equals 1000

See also The following Spreadsheet built-in functions:

ROUND

ROUNDUP

ROUNDDOWN

CEILING

FLOOR

RT_INDICATOR@

Implements the TIB directional indicator

Format RT_INDICATOR@(cell, up, down, no)

Arguments

cell	A cell address in the spreadsheet
up	The character displayed when the value in the cell has increased. This character is converted to a zaph dingbat. This character is typically "ö".
down	The character displayed when the value of the cell has decreased. This character is converted to a zaph dingbat. This character is typically "ô".
no	The character displayed when the cell has been updated but the value of the cell has not changed. This character is converted to a zaph dingbat. This character is typically "õ".

Description This function allows you to display a character in the spreadsheet that indicates whether the value in a cell increased, and decreased or stayed the same when the cell was last updated.

You can control the color of the displayed characters by using the Spreadsheet built-in functions to check for the character and change the color appropriately. An example of an RT_INDICATOR@ cell entry follows:

```
+RT_INDICATOR@(B1,"ö","ô","õ")&IF(CURRENT()="ô",ATTRS("color","Red"))&IF(CURRENT()="ö",ATTRS("color","Blu-
e"))&IF(CURRENT()="õ",ATTRS("color","Green"))
```

RUN@

Executes a macro

Format RUN@(args)

Arguments

arg[0]	The number of times to run a macro.
arg[1]	The name of the macro being run.
arg[2]...arg[7]	Arguments to the macro being run.

Description Executes a macro a number of times. This macro takes an array as an argument. If arg[0] = 1, the macro is run in the same way as if it were run using NEW_TASK@.

RUN_PROGRAM@

Executes a shell command-line program

Format status = RUN_PROGRAM@(command, outputFile[,errorFile])

Arguments

command	A program that can be executed from the shell from which Applix*ware* is invoked. Pipes (|) and wildcards are not supported.
outputFile	The path name to which the result of command is being directed.
errorFile	The path name to which the UNIX error message, if any, is being directed. The creation of errorFile implies that an error has occurred even if errorFile is empty.

Description Executes a specified program and directs the output and exit status to separate files. If the command fails, RUN_PROGRAM@ returns a number (usually 127). If the command is successful, RUN_PROGRAM@ returns an exit status of 0. The program must be in your UNIX search PATH. Otherwise, code 22 is returned.

RUN_PROGRAM@ is similar to **SHELL_COMMAND@**. The differences are:

· RUN_PROGRAM@ directs its result to a file, whereas SHELL_COMMAND@ stores the result in a system variable.

· RUN_PROGRAM@ traps errors; SHELL_COMMAND@ does not.

· SHELL_COMMAND@ supports pipes (|) and wildcards; RUN_PROGRAM@ does not. (RUN_PROGRAM@ does not start a C shell.)

See also **RUN_PROGRAM_UNPENDED@**
RUN_PROGRAM_WITH_INPUT@

RUN_PROGRAM_PID@

Executes a shell program and returns that program's PID

Format pid = RUN_PROGRAM_PID@(programName)

Arguments programName

Description Runs a shell program. The PID (Program ID) for this shell program is returned. See **RUN_PROGRAM@** for more information.

RUN_PROGRAM_UNPENDED@

Executes a command-line program in the background

Format status = RUN_PROGRAM_UNPENDED@(command, outputFile[, errorFile])

Arguments

command	A program that can be executed from the shell from which Applix*ware* is invoked. Pipes (|) and wildcards are not supported.
outputFile	The path name to which the result of command is being directed.
errorFile	The path name to which the UNIX error message, if any, is being directed. The creation of errorFile implies that an error has occurred, even if errorFile is empty.

Description RUN_PROGRAM_UNPENDED@ differs from RUN_PROGRAM@ in that RUN_PROGRAM_UNPENDED@ does not interrupt any other Applix*ware* processes. It blocks only the macro that calls it.

If the command fails, RUN_PROGRAM_UNPENDED@ returns a number (usually 127). If the command is successful, this macro returns an exit status of 0. The program must be in your UNIX search PATH. Otherwise, code 22 is returned.

See also RUN_PROGRAM_WITH_INPUT@

RUN_PROGRAM_WITH_INPUT@

Executes a command-line program using input from a file

Format status = RUN_PROGRAM_WITH_INPUT@(command, inputFile, outputFile)

Arguments

command	A program that can be executed from the shell from which Applix*ware* is invoked. command can be an operating system command or any other executable file that takes standard input. Pipes (|) and wildcards are not supported.
inputFile	The standard input for command.
outputFile	The path name to which the result of command is being directed.

Description Runs a specified program using inputFile, and directs the result to outputFile. If the program fails, RUN_PROGRAM_WITH_INPUT@ returns a number (usually 127). If the program is successful, RUN_PROGRAM_WITH_INPUT@ returns an exit status of 0. The program must be in your UNIX search PATH. Otherwise, code 22 is returned.

See also RUN_PROGRAM@
RUN_PROGRAM_UNPENDED@

RUN_PROGRAM_WITH_INPUT_UNPENDED@

Runs an external program

Format status=RUN_PROGRAM_WITH_INPUT_UNPENDED@ (command, inputFile, outputFile, errorFile)

Arguments

command	A program that can be executed from the shell from which Applix*ware* is invoked. command can be an operating system command or any other executable file that takes standard input. Pipes (|) and wildcards are not supported.

inputFile	The standard input for command.
outputFile	The path name to which the result of command is being directed.
errorFile	The path name to which the UNIX error message, if any, is being directed. The creation of errorFile implies that an error has occurred, even if errorFile is empty.

Description Runs a specified program using inputFile, and directs the result to outputFile.

If the program fails, RUN_PROGRAM_WITH_INPUT_UNPENDED@ returns a number (usually 127). If the program is successful, it returns an exit status of 0. The program must be in your UNIX search PATH. Otherwise, code 22 is returned.

When this macro runs, the only Applix*ware* or ELF process that is blocked is that macro that calls it.

See also RUN_PROGRAM@

RUN_PROGRAM_UNPENDED@

SAVE_CHECK@

Checks if a file is writable

Format flag = SAVE_CHECK@(filename)

Arguments filename The name of the file being checked.

Description Checks if a file is writable. If it is not, it tries to make the file writable. If the file can be made writable, TRUE is returned. Otherwise, this function returns FALSE.

See also FILE_WRITABLE@

SAVE_PREFERENCES@

Saves the ``ax_prof4'' Applix*ware* preferences file

Format SAVE_PREFERENCES@([filename])

Arguments filename Specifies the file to which the preferences are written, it this argument is NULL, the preferences are written to ax_prof4.

Description Normally, ax_prof4 is not saved until Applix*ware* is exited. SAVE_PREFERENCES@ saves the ax_prof4 file when it is executed.

Example

See also CHANGE_PREFS@
EDITPREFS@

SAVE_PROMPT@

Displays a Save As dialog box and returns the pathname of the file selected

Format SAVE_PROMPT@(format doc_format_ docinfo, types, orig_name, help_topic, nocheck, export, app_id, app_toggle1, sfxarr, app_toggle2, flag1, flag2, dir1, orig_writable, winTitle, db_cb_info)

Arguments

docinfo	The initial file name and permissions.
types	Strings for export options that are placed into the export scrolling list (visible only if the export flag is set to TRUE). The index of the selected format will be placed in the save_mode member of the returned doc_format_ structure.
orig_name	The original path of the file being saved.
help_topic	The name assigned to the help text for this dialog box. This name is treated the same way as any other dialog box name within the Help system.
nocheck	If set to TRUE, no warning message will be generated if the file already exists.
export	If set to TRUE, export behavior (the scrolling list of types, the alternate extension, etc.) will be used.
app_id	The called ID (e.g., APP#WORDS_).
app_toggle1	The optional intial value of application-specific toggle 1.
sfxarr	An array of suffixes, parallel to types. If it is NULL, the suffixes will be automatically generated from filter.sp (export only).
app_toggle2	The optional intial value of application-specific toggle 2.
flag1	an application-specific control field.
flag2	an application-specific control field.

dir1	The directory name whose contents are displayed in the Save As dialog box list box. If dir1 is not included, the current directory's files are displayed.
orig_writable	If set to TRUE, no warning message will be displayed when overwritting orig_name, regardless of the value of no_check. Set to FALSE if saving a read-only document.
winTitle	The title used for the Save As dialog box.
db_cb_info	Dialog callback information, for use in the db_dialog_callbacks@ macro.

Description Displays the Save As dialog box with its list box containing files meeting the attributes you specify. The Save As dialog box includes a list box, a file matching criteria entry box, a Directory option button; file permission option buttons; and Save, Cancel, and Help push buttons. The following doc_format_ structure is returned (as defined in fileinfo_.am):

```
format doc_format_
        name,                   ' Pathname of doc
        docid,                  ' Doc's unique filing identifier (while app is running)
        on_disk,   ' TRUE if file has been saved/read under this name
        save_mode,              ' 0-Normal, 1-Portable, 2...-special format or export format index
        grp_access,             ' 0-none, 1-read, 2-write
        all_access,             ' 0-none, 1-read, 2-write
        writeable, ' TRUE if this user has write access
        orig_rev,  ' revision of original file before conversions
        non_native              ' TRUE if the file was imported
```

SAVE_VERSION_CHECK@

Returns Boolean indicating if a version mismatch

Format flag = SAVE_VERSION_CHECK@(rev)

Arguments

rev	The document's version number, which is the orig_rev component of the doc_format_ format (contained in fileinf_.am in your elf sub-directory). This is the version number of the software that saved the document; it is not a revision/version number for the document itself.

Description Tests rev against the current version number when saving a document. If a mismatch occurs and the user's profile requests a warning to occur, a dialog box is displayed and the user is asked if the document should be saved. FALSE is returned if the document is not saved. TRUE is returned if the user saves the document or has set a profile indicating that no warning should appear.

SCHEME_TEMPLATE_DIR@

Returns the color scheme template directory

Format SCHEME_TEMPLATE_DIR@()

Description Returns the color scheme template directory used by Applixware Presents. Color schemes are collections of colors that can be re-used across multiple presentation documents. By default, this macro returns the directory /*install_dir*/templates/scheme, where install_dir is your Applixware install directory.

SECOND@

Extracts the second value from a serial time number

Format SECOND@(timeNumber)

Arguments timeNumber A serial time number.

Description SECOND@ is a time function which extracts the second (0-59) value from a serial time number. You can enter a serial time number as an argument for the SECOND@ function. A formula that contains the time serial number 0.7757176593 (which represents a time of 18 hours, 37 minutes and 2 seconds) returns 2.

You can also use the **NOW@** or **TIME@** functions as arguments in the SECOND@ function. For example, the formula SECOND@(TIME@(10,20,30)) returns 30.

SELECT_WINDOW@

Activates an Applix*ware* window and optionally raises the window to the foreground

Format SELECT_WINDOW@(id[, foreground])

Arguments

id	A large or small window id for an apllication window, or a large window id for a dialog window. The value is the same number returned by CURRENT_WINDOW_NUM@.
foreground	Indicates whether the activated window should be promoted to the foreground. If foreground is TRUE, the activated window is raised above all other windows; if foreground is the default FALSE setting, the activated window is not raised or lowered from its current position.

Description Makes the specified window the active window. Use this macro whenever a task switches from one window to another. For example, if you open a Words window, write a file and then switch to a Spreadsheet window, you need to establish that new Spreadsheets window as the current task.

Example

See also CURRENT_WINDOW_NUM@

LIST_OF_WINDOWS@

SELECT_WINDOW_BY_NAME@

WINDOW_INFO@

SELECT_WINDOW_BY_NAME@

Makes an Applix*ware* window the active window

Format SELECT_WINDOW_BY_NAME@(name)

Arguments name — The title (a string) of the window you want to be the active window.

Description Makes the window with the specified title the active window. If the window does not exist, SELECT_WINDOW_BY_NAME@ displays a dialog box that lists the names of all current windows so that you can choose a window to display. SELECT_WINDOW_BY_NAME@ does not promote a window to the foreground; you must use SELECT_WINDOW@ to promote an application window.

See also LIST_OF_WINDOWS@

SELECT_WINDOW@

SET_COLORS@

Sets the color attributes for Applix*ware* windows

Format SET_COLORS@ (workForeRed, workForeGreen, workForeBlue, workBackRed, workBackGreen, workBackBlue, foreRed, foreGreen, foreBlue, backRed, backGreen, backBlue)

Arguments workForeRed

The red saturation for the foreground color in a window's work area.

workForeGreen	The green saturation for the foreground color in a window's work area.
workForeBlue	The blue saturation for the foreground color in a window's work area.
workBackRed	The red saturation for the background color in a window's work area.
workBackGreen	The green saturation for the background color in a window's work area.
workBackBlue	The blue saturation for the background color in a window's work area.
foreRed	The red saturation for the foreground color in a window's control panel areas.
foreGreen	The green saturation for the foreground color in a window's control panel areas.
foreBlue	The blue saturation for the foreground color in a window's control panel areas.
backRed	The red saturation for the background color in a window's control panel areas.
backGreen	The green saturation for the background color in a window's control panel areas.
backBlue	The blue saturation for the background color in a window's control panel areas.

Description Sets the background and foreground color attributes for Applix*ware* windows. Both control panel colors and work area colors can be specified. Specify a number from 0 (no color) to 255 (pure color). The background and foreground colors are a result of the combination of the red, green, and blue color saturations.

· If you set the red, green, and blue saturations to 255 for a background or foreground color, the color displayed is white.

· If you set the red, green, and blue saturations to 0 for a background or foreground color, the color displayed is black.

· If you do not want to change a value for one of the attributes, specify NULL for the argument.

NOTE: On some displays, setting the red, green, and blue values to 0 displays white and setting the red, green, and blue values to 255 displays black.

SET_CURRENT_DIR@

Makes the directory you specify the current directory

Format SET_CURRENT_DIR@(name)

Arguments	name	The full path name of the directory you want to make the current directory.

Description Changes the current directory to the directory specified by name.

Example

See also CURRENT_DIR@

SET_LOOK_FEEL_VALUES@

Sets look-and-feel attributes

Format SET_LOOK_FEEL_VALUES@ (volume, click, toggles, width, accelFlag, pointerStyle, hourglassStyle, dialogsFlag, dimension, iconSize[, mnemonicsFlag[, wmFocusFlag]])

Arguments	volume	The bell volume. volume can be a number from 0 (no volume) to 255 (highest volume).
	click	Double-click time. Indicates the maximum amount of time allowable between clicks in order for two clicks to be interpreted as a double-click. Specify a number in milliseconds.
	toggles	Indicates the display used for toggle menu items when they are active. 0 check mark 1 toggle button
	width	The menu width: 0 always make room for check marks 1 make room only if needed
	accelFlag	Indicates whether to show accelerator keys in pull-down menus. Specify TRUE to display accelerators, FALSE to not display accelerators.
	pointerStyle	Indicates the symbol used for the mouse pointer.

hourglassStyle	Indicates the symbol used for the mouse pointer when the machine is in a time waiting state.
dialogsFlag	Indicates whether to use full decorations around dialog boxes. Specify TRUE to use full decorations, FALSE to not use full decorations.
dimension	Indicates whether to display dialog box controls as 3-dimensional or as 2-dimensional. Specify 0 to display as 3-dimensional, 1 to display as 2-dimensional.
iconSize	The set-aside icon size. The possible values are: 16x16, 32x32, 50x50, 64x64. This attribute is not changed until you re-start Applix*ware*.
mnemonicsFlag	Indicates whether to display the menu options' mnemonics. Set to TRUE to display the mnemonics; set to FALSE to suppress the mnemonics. If set to TRUE, an underscore appears below each mnemonic on the menu options.
wmFocusFlag	A Boolean value if set to TRUE indicates that window termination policy is determined by your window manager. If your window manager uses click-to-type rather than focus-follows-pointer you should set this value to TRUE. Set this value to FALSE if your window manager uses the focus-follows-pointer policy. This is the default

Description Changes the look-and-feel attributes to the values you specify. If you do not want to change a value for one of the attributes, specify NULL for the argument.

Values must be supplied for volume and click or the values will be interpreted as 0 (no bell volume and no double-click).

The default volume value is 70; the default click value is 500.

To change color look-and-feel attributes, use **SET_COLORS@**.

Example

SET_MACRO_PARENT_TASK@

Changes the task's master

Format flag = SET_MACRO_PARENT_TASK@(id)

Arguments

id	The id of the task that will becomes the current task's parent. If id has a value of 0, the current task is made the top-level task,

Description Changes the current task's parent to the one identified by id.

If the scheduler loop of axmain is entered and there are no top level tasks running, the Applix*ware* axmain process exists. If there is a request for axmain to exit and there are top level tasks running, the axmain process does not exist until the top level task has also existed.

All Applix*ware* applications (Words, Graphics, Spreadsheets, Data, and Mail) are top level tasks. If any application window is up or if Mail is conducting a mail transaction, axmain does not exit.

All ELF-based applications should execute the following macro:

```
SET_MACRO_PARENT_TASK@(0)
```

This insures that the macro and its dialog boxes are not slaved to a C-based application. It also causes the ELF task's child macro tasks to return this task number if they invoke **MACRO_PARENT_TASK@**.

The behavior of ELF task is indeterminate unless explicitly specified. Each ELF task must declare whether it wants to be a top level or non-top level task before it performs any operation which causes it to relinquish control and return to the scheduler. This declaration is made as follows:

```
        'Establish task as a top level task
SET_MACRO_PARENT_TASK@(0)
```

or

```
        'Establish task as non-top level task
SET_MACRO_PARENT_TASK@(1)
```

The following macros can be used so that the macro launching the new task can control the status of the child:

```
FUNCTION NEW_MAJOR_TASK(name, a1, a2, a3, a4, a5, a6, a7, a8)
    return ( NEW_TASK@(``START_MAJOR_TASK'', name, a1, a2, a3, a4, a5, a6,
            a7, a8))
ENDFUNCTION

FUNCTION START_MAJOR_TASK(name, a1, a2, a3, a4, a5, a6, a7, a8)
    SET_MACRO_PARENT_TASK@(0)
    return ( !name( a1, a2, a3, a4, a5, a6, a7, a8))
ENDFUNCTION

FUNCTION NEW_MINOR_TASK(name, a1, a2, a3, a4, a5, a6, a7, a8)
    return ( NEW_TASK@(``START_MINOR_TASK'', name, a1, a2, a3, a4, a5, a6,
            a7, a8))
ENDFUNCTION

FUNCTION START_MINOR_TASK(name, a1, a2, a3, a4, a5, a6, a7, a8)
    SET_MACRO_PARENT_TASK@(1)
```

```
        return ( !name( a1, a2, a3, a4, a5, a6, a7, a8))
ENDFUNCTION
```

SET_MACRO_TOP_LEVEL@

Makes a task top level

Format SET_MACRO_TOP_LEVEL@(makeTopLevelFlag)

Arguments makeTopLevelFlag

A Boolean value which if set to TRUE tells Applix*ware* that it should make this task the top level task.

See also **SET_MACRO_PARENT_TASK@.**

SET_MENU_BAR_ID@

Replaces a menu bar

Format SET_MENU_BAR_ID@(id)

Arguments id — A unique number identifying the menu bar for the dialog box or application window. id can be in the range of 21 to 499:

21-99	Reserved for dialog boxes.
100-199	Reserved for Spreadsheet menu bar.
200-299	Reserved for Words menu bar.
300-399	Reserved for Graphics menu bar.
400-499	Reserved for Macro editor menu bar.

Description Changes a window's menu bar to the menu bar identified by id. (This id is set using the **SET_SELECTIONS@** macro.)

SET_SELECTIONS@

Defines a custom menu bar definition

Format SET_SELECTIONS@(id, array)

Arguments id — A unique number identifying the custom menu bar for the dialog box or application window. id can be in the range of 21 to 499:

21-99	Reserved for dialog boxes.

100-199	Reserved for Spreadsheet menu bar.
200-299	Reserved for Words menu bar.
300-399	Reserved for Graphics menu bar.
400-499	Reserved for Macro editor menu bar.

id cannot be a number between 0 and 20, as those numbers are hard-coded for standard Applix*ware* menu bars listed in the app_ids_.am file in your axlocal/elf directory.

array	Name of the array containing the menu bar definition.

Description Stores a menu bar definition specified by array so that you can impose that menu bar on a dialog box or application window. SET_SELECTIONS@ is not appropriate for modifying default menu bars (ax_ss4, ax_wp4, ax_gr4, and ax_me4). If you want to modify a default menu bar, use **UPDATE_SELECTIONS_FILE@**.

To display a custom menu bar in a dialog box, SET_SELECTIONS@ is followed by **DB_MENU_BAR@**. To display a custom menu bar in a new application window, SET_SELECTIONS@ is followed by one the following:

- **OPEN_DOC@**
- **GR_APPLICATION_DLG@**
- **ME_APPLICATION_DLG@**
- **SS_APPLICATION_DLG@**
- **WP_APPLICATION_DLG@**

To display a custom menu bar in the current application window, use **SET_MENU_BAR_ID@** after calling SET_SELECTIONS@. To store into memory the *current* menu bar, make a call only to SET_SELECTIONS@.

SET_SUBSTRING@

Replaces a substring with a new text string

Format newString = SET_SUBSTRING@(origString, start, span, repString)

Arguments

origString	The string of characters, a portion of which is replaced by SET_SUBSTRING@.
start	The starting character position of the substring within origString.
span	The number of characters in the substring.
repString	The substring replacing the old substring in string.

Description Creates newString based on another string origString where part of origString has been changed to repString. To find the starting character position of repString, use STRING_INDEX@.

See also REPLACE_SUBSTR@

SUBSTRING@

SET_SYSTEM_VAR@

Assigns a value to a system variable

Format SET_SYSTEM_VAR@(name, value)

Arguments

name	The name, a string, of the system variable to which you want to assign a value.
value	The value to assign to the system variable. value may be a string, number, or array.

Description Assigns a value to the system variable specified by name. The system variable is global to all tasks performed during the Applix*ware* session in which it is defined.

See also SYSTEM_VAR@

SHELL_COMMAND@

Executes a shell command and returns the output as a string array

Format array = SHELL_COMMAND@(command)

Arguments

command	An operating system command. If your operating system command language allows it, you can string multiple commands together as you would at an operating system prompt.

Description Returns a string array representing the output of the specified operating system command(s).

Example

See also RUN_PROGRAM@

RUN_PROGRAM_WITH_INPUT@

RUN_PROGRAM_UNPENDED@

NOTE: This macro is an obsolete version of RUN_PROGRAM@.

SIN@

Returns the trigonometric sine of an angle based on it radian value

Format value = SIN@(value)

Arguments	value	The degree of an angle. value must be a number from -1 to 1.

Example

See also **COS@**
LOG@
LOG10@
SQR@

SLIDE_TEMPLATE_DIR@

Returns the slide template directory

Format SLIDE_TEMPLATE_DIR@()

Description Returns the 35mm template directory used by Applixware Presents. This directory contains templates used to create 35 millimeter film slides. By default, this macro returns the directory /*install_dir*/templates/35mm, where *install_dir* is your Applix*ware* install directory.

SNAPXY@

Creates a snapshot that is xSize by ySize pixels

Format SNAPXY@(xSize, ySize)

Arguments	xSize	The width of the snapshot in pixels.
	ySize	The height of the snapshot in pixels.

Description Creates a rectangular box that is xSize pixels wide by ySize pixels high. After the rectangle is created, you can place the box anywhere on the screen using your mouse. After the box is placed, clicking the left mouse button tells Applix*ware* to capture the region.

See also SNAP_TO_CLIPBOARD@

SNAP_TO_GR@

SNAP_TO_GRXY@

SNAP_TO_CLIPBOARD@

Grabs an image and places it in clipboard

Format SNAP_TO_CLIPBOARD@([grabServerFlag[, delayTime]])

Arguments

grabServerFlag
: A Boolean value which if set to TRUE tells the X Server to perform an XGrabServer call. (Grabbing the server means that only requests by the calling client are acted on. All other requests are queued until the snapshot is made.)

delayTime
: The time between starting the macro and the capturing of the image. Use this argument to draw a transitory image on the screen.

Description Places a *snapshot* of the image within a rectangle into the clipboard.

See also SNAPXY@

SNAP_TO_GR@

SNAP_TO_GRXY@

SNAP_TO_GR@

Creates a snapshot and places it into a file

Format SNAP_TO_GR@(outfile[, grabserverFlag[, delaytime]])

Arguments

outfile
: The file into which the snapshot is written.

grabserverFlag
: A Boolean value which if set to TRUE tells the X Server to perform an XGrabServer call. (Grabbing the server means that only requests by the

	calling client are acted on. All other requests are queued until the snapshot is made.)
delayTime	The time between starting the macro and the capturing of the image. Use this argument to draw a transitory image on the screen.

See also SNAPXY@

SNAP_TO_CLIPBOARD@

SNAP_TO_GRXY@

SNAP_TO_GRXY@

Creates a snapshot and places it in a file

Format SNAP_TO_GRXY@(outfile, xSize, ySize)

Arguments

outfile	The file to which the snapshot is written.

Arguments

xSize	The width of the snapshot in pixels.
ySize	The height of the snapshot in pixels.

Description Creates a rectangular box that is xSize pixels wide by ySize pixels high. After the rectangle is created, you can place the box anywhere on the screen using your mouse. After the box is placed, clicking the left mouse button tells Applix*ware* to capture the region.

See also SNAPXY@

SNAP_TO_CLIPBOARD@

SNAP_TO_GR@

SORT@

Sorts an array of strings or numbers in descending or ascending order

Format newArray = SORT@(array, descendFlag)

Arguments

array	The string or number array to be sorted. If array containing both strings and numbers is supplied, an error is thrown.
descendFlag	A Boolean value which if set to TRUE indicates that the array is sorted in descending order (z-a). If this argument is FALSE, array is sorted in ascending order (a-z).

Description Returns a sorted array of values. The array to be sorted must either contain all strings or all numbers; it cannot contain both . You cannot sort an array containing arrays. That is, multi-dimensional sorts are not allowed. If array is NULL, NULL is returned.

Example

See also ARRAY_APPEND@
ARRAY_TRANSPOSE@
FORMAT_ARRAY@
SORT_INDEXED@

SORT_INDEXED@

Returns a sorted array and information about the sort

Format array = SORT_INDEXED@(toSort, descFlag, caseFlag)

Arguments

toSort	The array being sorted.
descFlag	A Boolean value which if set to TRUE indicates that the array is sorted in descending order (z-a). If this argument is FALSE, then array is sorted in ascending order (a-z).
caseFlag	A Boolean value which if set to TRUE indicates that the case of letters is ignored when the array is sorted.

Description Sorts an array and provides information about the sort order. This macro lets you simulate a multi-dimensional sort. (ELF has the restriction that it cannot sort multi-dimensional arrays.) Specifically, it returns an array of arrays as follows:

· The first array contains the sorted information. If the input array is called IN, the sorted equivalent to it can be found in OUT[0].

· The second array OUT[1] tells "where the input data ended up". That is, for any element in the input array IN, the corresponding element in OUT[1] indicates the position to which it was sorted. For example, if element IN[3] were sorted into position OUT[0,15], the value of OUT[1,3] would be 15.

· The third array OUT[2] tells "where the sorted data came from". That is, if you look at any element in OUT[0,i], the corresponding element OUT[2,i] tells you its original array position. For example, assume the tenth item in the input array IN was moved to the fifth element in the output array OUT[0] . That is, OUT[0,5] = IN[10]. The value in item OUT[2,5] array would be 10, indicating that the tenth item of the input array was moved to this fifth array position. That is:

```
    OUT[0,5] = IN[OUT[2,5] ]
    In more general terms:
    OUT[0, i] = IN[OUT[2,i] ]
```

Here is how you would use this macro. Suppose you have the following format:

```
format myFormat
        first.
        second
```

Within some macro, you might have the following declarations and data:

```
macro foo
    var format arrayof myFormat stuff
    ...
    stuff[0].first = 99
    stuff[0].second = "z"
    stuff[1].first = 54
    stuff[1].second = "m"
    stuff[2].first = 67
    stuff[2].second = "b"
    ...
    stuff = sort_two_element_format(stuff)
    ....
endmacro
```

Here is the general macro that will sort any two-item format:

```
macro sort_two_element_format(format arrayof myFormat stuff)
    var format arrayof myFormat newStuff
    var sorts
    var i

                'Extract the first column of the input
                'format, which is actually an array, then
                'sort it
    sorts = sort_indexed@(array_column@(stuff,0))
                'Now move original elements into
                'their new positions
    for i = 0 to array_size@(stuff) - 1
        newStuff[i] = stuff[sorts[2,i]]
    next i
    return (newStuff)
endmacro
```

In a similar manner, you could also sort a multi-dimensional array.

The SORT_INDEXED@ macro determines if a string or numeric sort is used based on the first element of the passed array. All elements must be of the same type.

See also SORT@

SPLIT_PATHNAME@

Converts path name to directory/file name

Format pathArray = SPLIT_PATHNAME@(pathname)

Arguments

pathname	The path name to be divided into a directory name and a file name.

Description Returns a two-element array containing the directory name and the file name of a pathname. The directory name is in the first element of pathname; the file name is in the second element.

See also PARSE_PATHNAME@

Start Inset Macros

Opens an inset

Format *START_INSET_MACRO_NAME*@(pathName, rootDocPath, winTitle, readOnly)

Arguments

pathName	The name of a file in the current file system containing inset data.
rootDocPath	The name of the original file referenced by the inset. This name is used to resolve relative links in the inset.
winTitle	The title of the window that appears when the inset file is displayed in an Applix*ware* document.
readOnly	If TRUE, the file is opened in Read Only mode. If FALSE, the file is opened in Read/Write mode.

Description Start inset macros import the file specified by rootDocPath into an appropriate Applix*ware* document type. The macro you need to use depends on the type of inset you are trying to open. The following lists the Start Inset macros, along with the file type to which the macro applies.

AUDIO_START_INSET@ plays an Audio inset.

BITMAP_START_INSET@ opens a Bitmap Editor inset.

CHART_START_INSET@ opens a chart inset with Applixware Graphics.

DBASE_START_INSET@ opens a Data Query inset with Applixware Data.

EQN_START_INSET@ opens an Equation Editor inset.

HTML_START_INSET@ opens an HTML inset with the HTML Author.

ME_START_INSET@ opens a Macro Editor inset.

SLIDESHOW_START_INSET@ opens an Applixware Presents Slideshow inset with Applixware Presents.

SS_START_CSV_INSET@ opens an ASCII file in comma separated values form with Applixware Spreadsheets.

SS_START_DIF_INSET@ opens a Data Interchange Format inset with Applixware Spreadsheets.

SS_START_INSET@ opens an Applixware inset. The target document can be generated with Words, Spreadsheets, Data, or Graphics. Spreadsheets determines which document type the target document is, and opens the inset with that document type. For example, a Graphics inset would be opened in Applixware Graphics.

SS_START_SYLK_INSET@ opens a SYLK inset in Applixware Spreadsheets.

SS_START_WK3_INSET@ opens a LOTUS 1-2-3 WK3 inset with Applixware Spreadsheets.

SS_START_WK4_INSET@ opens a LOTUS 1-2-3 WK4 inset with Applixware Spreadsheets.

SS_START_WKS_INSET@ opens a LOTUS 1-2-3 WK1 inset with Applixware Spreadsheets.

SS_START_XLS_INSET@ opens a Microsoft Excel inset with Applixware Spreadsheets.

START_CGM_INSET@ opens a CGM inset with Applixware Graphics.

START_DXF_INSET@ opens a DXF inset with Applixware Graphics.

START_EPS_INSET@ opens an EPS inset with Applixware Graphics.

START_FAX_INSET@ opens a group 3 FAX inset with Applixware Graphics.

START_GEM_INSET@ opens a GEM inset with Applixware Graphics.

START_GIF_INSET@ opens a GIF inset with Applixware Graphics.

START_GR_INSET@ opens an Applixware Graphics inset.

START_HPGL_INSET@ opens an HPGL inset with Applixware Graphics.

START_ILBM_INSET@ opens an ILBM inset with Applixware Graphics.

START_IRIS_INSET@ opens a Silicon Graphics IRIS inset in Applixware Graphics.

START_JPEG_INSET@ opens a JPEG inset with Applixware Graphics.

START_MPNT_INSET@ opens a MacPaint inset with Applixware Graphics.

START_MSWB_INSET@ opens a Microsoft Windows Bitmap inset with Applixware Graphics.

START_PBM_INSET@ opens a Portable Bitmap inset in Applixware Graphics.

START_PCX_INSET@ opens a PCX inset in Applixware Graphics.

START_PGM_INSET@ opens a Portable Graymap inset in Applixware Graphics.

START_PICT_INSET@ opens a MacDraw PICT inset in Applixware Graphics.

START_PICT2_INSET@ opens a MacDraw PICT2 inset in Applixware Graphics.

START_PPM_INSET@ opens an X11 Portable Pixmap inset in Applixware Graphics.

START_PPT_INSET@ opens a PPT inset in Applixware Presents.

START_RS_INSET@ opens a Sun Raster inset in Applixware Graphics.

START_TGA_INSET@ opens a TGA inset with Applixware Graphics.

START_TIF_INSET@ opens a TIFF inset with Applixware Graphics.

START_WMF_INSET@ opens a Window metafile inset in Applixware Graphics.

START_WPG_INSET@ opens a WPG inset with Applixware Graphics.

START_XBM_INSET@ opens an X Windows Bitmap inset in Applixware Graphics.

START_XPM_INSET@ opens an X Windows Pixmap inset in Applixware Graphics.

START_XWD_INSET@ opens an X Windows Dump inset in Applixware Graphics.

WP_START_ASCII_LINES_INSET@ opens an ASCII inset with Applixware Words. Each line in the ASCII file is a separate paragraph in the Words file.

WP_START_ASCII_PARAS_INSET@ opens an ASCII inset with Applixware Words. Paragraph breaks in the Words file occur when lines of text are separated by two RETURNS in the target file.

WP_START_INSET@ opens an Applixware inset. The target document can be generated with Words, Spreadsheets, Data, or Graphics. Words determines which document type the target document is, and opens the inset with that document type. For example, a Graphics inset would be opened in Applixware Graphics.

WP_START_MACWORD50_INSET@ opens a Microsoft Word for the Macintosh, version 5.0 inset.

WP_START_MSWORD_INSET@ opens a Microsoft Word for DOS inset. The target file must have been created with version 5.5 or earlier of Microsoft Word.

WP_START_MSWORD70_INSET@ opens a Microsoft Word inset. The target file must have been created with version 7.0 of Microsoft Word for Windows, or version 6.0 of Microsoft Word for Macintosh.

WP_START_RTF_INSET@ opens a Rich Text Format inset.

WP_START_WINWORD60_INSET@ opens a Microsoft Word for Windows version 6.0 inset in Applixware Words.

SQR@

Returns the square root of a positive number

Format num = SQR@(value)

Arguments	value	A numeric expression, which must be a positive number.

Example

See also COS@
LOG@
LOG10@
SIN@

START_RECORD@

Turns on learn mode

Format START_RECORD@()

Description Places Applix*ware* into learn mode where actions you perform are recorded so that you can create keyboard macros. This is normally bound to the !F8 key. Use the STOP_RECORD@ macro to end learn mode. (This is dynamically bound to the !F8 after you enter learn mode.)

See also STOP_RECORD@

STOP_RECORD@

Exits learn mode

Format STOP_RECORD@()

See also START_RECORD@

STRING_INDEX@

Returns the starting character position of a substring

Format pos = STRING_INDEX@(string, substring)

Arguments

string	The string in which you want to locate a sub-string.
substring	The substring you want to locate.

Description Returns the starting character position of a substring within a string. String positions are one-based. Thus, if substring starts at the third character in string, 3 is returned. If STRING_INDEX@ does not locate substring, it returns 0. If substring is NULL, STRING_INDEX@ returns 1.

Example

See also STRING_INDEX_REVERSE@
SUBSTRING@

STRING_INDEX_REVERSE@

Returns the last character position of a substring with a string

Format pos = STRING_INDEX_REVERSE@(string, substring)

Arguments

string	The text in which you which you wish to locate a substring.
substring	The text being looked for within string.

Description Returns the last character position within string at which substring exists. This macro differs from STRING_INDEX@ in that it begins looking from the end of the string rather than the beginning.

String positions are one-based. Thus, if substring starts at the third character in string, 3 is returned. If STRING_REVERSE_INDEX@ does not locate substring, it returns 0. If substring is NULL, STRING_REVERSE_INDEX@ returns a number 1 greater than the length of the string.

See also SUBSTRING@

STRING_TO_BINARY@

Converts an ELF string into a binary object

Format object = STRING_TO_BINARY@(string)

Arguments	string	The string to be converted to a binary data object.

Description Converts a a text string to a binary data object.

See also **BINARY_TO_STRING@**

STRING_TO_BIT7_ASCII@

Converts an external string into internal format

Format bytes = STRING_TO_BIT7_ASCII@(in, out, maxLen, format afile_info info)

Arguments	in	The string being converted.
	out	The converted string.
	maxLen	The maximum size of the output buffer.
	info	The encoding information or NULL. This is back slash character for escaping characters such as another back slash or the beginning of an encoded 8 bit or 16 bit value

Description Converts a string from its current representation, which can be 7-, 8-, or 16-bit into the internal Applix*ware* format, which is 7-bit ASCII.

The definition of afile_info is as follows:

```
format afile_info
        encoding,       'TRUE if the file is encoded
        version,        ' version number
        docType         'See recgfil_.am
```

STRING_TO_NUM@

Returns the numeric equivalent of an ASCII character

Format num = STRING_TO_NUM@(char)

Arguments	char	A string that is the character to convert into a number.

Description Returns a number corresponding to the ASCII code for the character specified.

Example

See also NUM_TO_STRING@
STRING_TO_VALUE@

STRING_TO_VALUE@

Converts a string to number

Format value = STRING_TO_VALUE@(num)

Arguments	num	A string containing a numeric value.

Description Converts a string containing numbers into its equivalent numeric value. This macro is most often used when communicating with non-ELF programs.

See also STRING_TO_NUM@

START_STROKE@

Indicates the starting position of a mouse pointer drag

Format START_STROKE@ (xpos, ypos[, button, shift, control])

Arguments	xpos	The pixel position, relative to the drawing area, of the mouse pointer on the x-axis.
	ypos	The pixel position, relative to the drawing area, of the mouse pointer on the y-axis.
	button	A number indicating the mouse button pressed during the pointer drag.
	shift	Indicates whether the SHIFT button is pressed while dragging.

button values:

0	left button
1	center button
2	right button

shift values:

1	SHIFT is pressed
0	SHIFT is not pressed

control	Indicates whether the CONTROL button is pressed while dragging.	
	1	CONTROL is pressed
	0	CONTROL is not pressed

Description Indicates the mouse pointer position at the beginning of a drag in a Graphics drawing area and specifies whether SHIFT or CONTROL is pressed during the drag.

See also STROKE_END@
STROKE_POINT@
STROKE_START@

STROKE_END@

Indicates the ending position of a mouse pointer drag

Format STROKE_END@ (xpos, ypos[,hvStatus])

Arguments

xpos	The pixel position, relative to the drawing area, of the mouse pointer on the x-axis.
ypos	The pixel position, relative to the drawing area, of the mouse pointer on the y-axis.
hvStatus	Only applicable to the line tool, this argument indicates whether a strictly horizontal or strictly vertical line was drawn. A hvStatus value of 1 indicates that a strictly vertical line was drawn. A hvStatus value of 2 indicates that a strictly horizontal line was drawn. This argument is only relevant to keystroke recordings involving STROKE_END@. If STROKE_END@ is used as part of a macro to specify the ending position of a stroke in Graphics, the hvStatus argument is not required.

Description Indicates the mouse pointer position at the end of a drag in a Graphics drawing area.

Example

See also START_STROKE@
STROKE_POINT@
STROKE_START@

STROKE_POINT@

Indicates mouse pointer position during a mouse drag

Format STROKE_POINT@(xpos, ypos)

Arguments

xpos	The pixel position, relative to the drawing area, of the mouse pointer on the x-axis.
ypos	The pixel position, relative to the drawing area, of the mouse pointer on the y-axis.

Description Indicates the mouse pointer position during a drag in a Graphics drawing area. It can be used to draw a free form object in a Graphics window by indicating the mouse pointer position as it is dragged.

Example

See also STROKE_END@
STROKE_POINT_BOUNDARY@
STROKE_START@
STROKE_START_EXTEND_SELECT@
STROKE_START_MULTI_SELECT@

STROKE_POINT_BOUNDARY@

Indicates a stroke boundary occurred

Format STROKE_POINT_BOUNDARY@(x, y)

Arguments

x	The point's x-coordinate.
y	The point`s y-coordinate.

Description Indicates that a stroke reached a window boundary. (A stroke is a movement of the mouse with a mouse button pressed.) In some cases, you will ignore this event as it means that you are not within the window's boundary. In some cases, however, you may want to scroll your window or perform some other action so to allow the stroking to continue.

See also STROKE_END@

STROKE_POINT@
STROKE_START@
STROKE_START_EXTEND_SELECT@
STROKE_START_MULTI_SELECT@

STROKE_START@

Sets the point that begins a stroke

Format STROKE_START@(x, y)

Arguments	x	The point's x-coordinate.
	y	The point`s y-coordinate.

Description Indicates the point at which a stroke movement begins. (A stroke is a movement of the mouse with a mouse button pressed.)

Example

See also STROKE_END@
STROKE_POINT@
STROKE_POINT_BOUNDARY@
STROKE_START_EXTEND_SELECT@
STROKE_START_MULTI_SELECT@

STROKE_START_EXTEND_SELECT@

Starts an extend stroke

Format STROKE_START_EXTEND_SELECT@(x,y)

Arguments	x	The point's x-coordinate.
	y	The point`s y-coordinate.

Description Indicates the point at which a extended-selection stroke movement begins. (An extended selection stroke is a movement of the mouse with a mouse button pressed and with the keyboard's shift key pressed.)

See also STROKE_END@

STROKE_POINT@
STROKE_POINT_BOUNDARY@
STROKE_START@
STROKE_START_MULTI_SELECT@

STROKE_START_MULTI_SELECT@

Starts a multi-select stroke

Format STROKE_START_MULTI_SELECT@(x, y)

Arguments

x	The point's x-coordinate.
y	The point`s y-coordinate.

Description Indicates the point at which a multi-selection stroke movement begins. (A multi selection stroke is a movement of the mouse with a mouse button pressed and with the keyboard's control key pressed.)

See also STROKE_END@
STROKE_POINT@
STROKE_POINT_BOUNDARY@
STROKE_START@
STROKE_START_EXTEND_SELECT@

SUBARRAY@

Returns an array beginning at a specified position

Format newArray = SUBARRAY@(array, start[, span])

Arguments

array	An array of arrays, a portion of which is returned by SUBARRAY@.
start	The starting character position in the specified array. Array positions are numbered from 0.
span	An optional number value that determines the number of array elements SUBARRAY@ returns.

Description Returns a part of an input array beginning at start. The number of array elements returned is specified using the span parameter. If span is omitted, an array is returned beginning at input array element start and continuing to the end of the input array.

If start's value is greater than the length of array, SUBARRAY@ returns a blank array. If the array element indicated by start or span is greater than the number of elements in the input array, the last array element returned is the last array element in the input array.

Assume that you have a 10-element array named foo.

foo1 = SUBARRAY@(foo,3,4)

This returns a 4-element array beginning at foo[3].

foo1 = SUBARRAY@(foo,3)

This returns a 7-element array beginning at foo[3].

foo1 = SUBARRAY@(foo,10,3)

NULL is returned.

See also SUBARRAY_REMOVE@

SUBARRAY_REMOVE@

Removes part of an array

Format newArray = SUBARRAY_REMOVE@(array, start, span)

Arguments

array	The array from which elements will be removed.
start	The starting array position in the specified array. Array positions are numbered from 0.
span	A value that determines the number of elements SUBARRAY_REMOVE@ returns.

Description Removes the portion of an array beginning at start and ending with span. This is a dangerous routine as it does not perform error checking on the array size.

See also ARRAY_DELETE@
LIST_REMOVE@
SUBARRAY@

SUBSTRING@

Returns the text string beginning at a specified position

Format substring = SUBSTRING@(string, start[, span])

Arguments

string	A string of characters, a portion of which is returned by SUBSTRING@.
start	The starting character position in the specified string. String character positions are numbered from 1. For example SUBSTRING@ ("Compilation failed", 13) returns a substring starting at the 13th character in the string "Compilation failed."
span	An optional number value that determines the number of characters SUBSTRING@ returns. For example, SUBSTRING@ ("Compilation failed", 13, 4) returns the string "fail". If you do not specify a value for span, SUBSTRING@ returns all characters from start to the end of the string.

Description If start's value is greater than the length of string, SUBSTRING@ returns a blank string. If span's value is greater than the length of string, SUBSTRING@ returns only the characters to the end of string.

Example

See also REPLACE_SUBSTR@
SET_SUBSTRING@
STRING_INDEX@

SUPPORTS_AUDIO@

Reports if audio is supported on a machine

Format flag = SUPPORTS_AUDIO@()

Description Returns TRUE if the current machine supports audio.

Example

SUPPRESS_ERROR_MESSAGES@

Suppresses and enables the posting of error messages

Format SUPPRESS_ERROR_MESSAGES@(flag)

Arguments	flag	Indicates whether to enable or disable the posting of error messages. TRUE suppresses the posting of error messages. FALSE (the default) causes error messages to be posted as usual.

Description Suppresses the posting of error messages whenever appropriate. Set it to TRUE whenever you run Applix*ware* in the background; otherwise, error messages *hang* the background process indefinitely. INFO_MESSAGE@ messages call for SUPPRESS_INFO_MESSAGES@.

Normally, you should set this macro to TRUE as soon as you invoke a background process, and turn it back off upon exiting the background process. Within a given macro, you can re-set this switch as often as necessary. You may also want to store error messages in variables.

NOTE: When SUPPRESS_ERROR_MESSAGES@ is TRUE, error conditions are not ignored. You must still include the appropriate error-handling code. The only difference is in the actual posting of error messages.

See also SUPPRESS_INFO_MESSAGES@

INFO_MESSAGE@

GR_APPLICATION_DLG@

SS_APPLICATION_DLG@

WP_APPLICATION_DLG@

SUPPRESS_INFO_MESSAGES@

Suppresses and enables the posting of error messages

Format SUPPRESS_INFO_MESSAGES@(flag)

Arguments	flag	Indicates whether to enable or disable the posting of info messages. TRUE suppresses the posting of info messages. FALSE (the default) causes info messages to be posted as usual.

Description Suppresses the posting of info messages whenever appropriate. Set it to TRUE whenever you run Applix*ware* in the background; otherwise, info messages *hang* the background process indefinitely. INFO_MESSAGE@ messages call for SUPPRESS_INFO_MESSAGES@.

Normally, you should set this macro to TRUE as soon as you invoke a background process, and turn it back off upon exiting the background process. Within a given macro, you can re-set this switch as often as necessary. You may also want to store info messages in variables.

NOTE: When SUPPRESS_INFO_MESSAGES@ is TRUE, error conditions are not ignored. You must still include the appropriate error-handling code. The only difference is in the actual posting of info messages.

See also SUPPRESS_ERROR_MESSAGES@
INFO_MESSAGE@
GR_APPLICATION_DLG@
SS_APPLICATION_DLG@
WP_APPLICATION_DLG@

SYS5@

Indicates if Applix*ware* is running on a UNIX System 5 machine

Format flag = SYS5@()

Description Returns TRUE if the machine on which Applix*ware* is installed is compiled as a UNIX System 5 machine; otherwise, it returns FALSE.

Example

SYSTEM_DIR@

Returns the name of the Applix*ware* installation directory

Format dir = SYSTEM_DIR@()

Description When Applix*ware* is installed, the Applix*ware* data files are placed in a directory. SYSTEM_DIR@ returns the full path name of the directory where the Applix*ware* data files reside.

Example

See also AXHOME_DIR@
SYSTEM_LANG_DIR@

SYSTEM_LANG_DIR@

Returns the name of directory containing language dependent files

Format dir = SYSTEM_LANG_DIR@()

Description Returns the name of the directory where language-dependent files are located. For example, this macro could return *install_dir*/axdata/lang.

Example

See also SYSTEM_DIR@

SYSTEM_VAR@

Returns the value of the system variable specified

Format value = SYSTEM_VAR@(name)

Arguments	name	A string that is the name of a system variable.

Description Returns the current value of the system variable specified by name.

See also SET_SYSTEM_VAR@

TABS_TO_SPACES@

Converts tabs to spaces in a line of ASCII text

Format newString = TABS_TO_SPACES@(string[, width])

Arguments	string	The text string to be converted. This is typically a string read from an operating system file.
	width	The optional tab width. The default value is 8 characters.

Description Used to display operating system command output within Applix*ware* in the same format as it is displayed at the operating system level.

Example

See also IS_WHITESPACE@

TASK_ID@

Returns the identification number of the task that calls TASK_ID@

Format id = TASK_ID@()

Arguments None.

Description Each Applix*ware* task is assigned a unique task identification number. TASK_ID@ returns the task ID of the task from which the macro is called.

See also KILL_TASK@
TASK_LIST@

TASK_LIST@

Returns an array of information about all current ELF tasks

Format infoArray =TASK_LIST@([caseFlag])

Arguments caseFlag If this optional flag is set to TRUE, task names are shown in uppercase.

Description Returns an array that gives the following information for each current ELF task:

id	The task ID number. This number can be passed to KILL_TASK@ to stop a task.
name	The name of the task. If the task was invoked using a macro, name is the name of the macro. If the task is an Applix*ware* C-based application, then the name of application is returned. For example, Applixware Words returns Words_.
topLevelFlag	If a task is a top-level task (a task that has called Set_Macro_Parent_Task@(0)), then this flag is set to TRUE(-1). If the task is not a top-level task, this flag is set to FALSE (0).

See also KILL_TASK@
TASK_ID@

TASK_TO_WINDOW@

Returns an active task's window ID

Format	windowID = TASK_TO_WINDOW@(taskID)	
Arguments	taskID	A task ID for an active task.
Description	Returns the small window ID for an active task. Returns -1 if the task is not active.	
See also	TASK_ID@ TASK_LIST@	

TASK_WINS_BUSY@

Sets or clears a task's hourglass

Format TASK_WINS_BUSY@(showHourglassFlag)

Arguments showHourglassFlag

A Boolean value which if set to TRUE tells Applix*ware* that it should display an hourglass cursor when the window cursor is over this task's window.

Description Sets or clears the hourglass icon that indicates that the task is busy. This hourglass is set for an individual task. That is, this hourglass will display for those windows created by the calling task.

Unlike the hourglass displayed when you use the ALL_WINDOWS_BUSY@ macro, this hourglass must be explicitly removed by calling this macro a second time with an argument of FALSE.

You should only use this function when a task is performing DELAY@ operations or using a communications channel. These operations tear down the hourglass used by ALL_WINDOWS_BUSY@.

TASKS

ELF supports both single-threaded and multi-threaded execution. Execution threads in ELF macros are called *tasks*.

The most simple ELF macros have a single task. You can start a new task using one of three ELF macros:

· PEND_FOR_NEW_TASK@ starts a new task, and waits for that task to complete before proceding with the next statement. The following macro starts a task that counts to 100. When the child task completes, the parent displays an INFO_MESSAGE@.

· NEW_TASK@ starts a task that executes until it gives up control of the thread. The child task can give up control of the thread by calling DELAY@ or DB_DISPLAY@.

The only difference between NEW_TASK@ and PEND_FOR_NEW_TASK@ is that tasks spawned with NEW_TASK@ can give up control of the thread, where tasks spawned with PEND_FOR_NEW_TASK@ must complete in order to give up control of the thread.

· NEW_TASK_UNPENDED@ starts a new task, and continues with the next statement. The new task is placed at the top of the scheduling queue. When the parent task completes, the ELF Scheduler determines which task in the queue is allocated time.

The ELF Scheduler maintains a queue of active tasks, and is responsible for allocating time to each ELF task.

See also NEW_TASK@, ELF Scheduler, PEND_FOR_NEW_TASK@, NEW_TASK_UNPENDED@

TEMP_DIR@

Returns the name of the Applix*ware* temporary directory

Format dir = TEMP_DIR@()

Description Returns the name of the directory in which Applix*ware* places temporary files. (Some Applix*ware* software needs to write information to disk on a temporary basis.)

Example

See also SYSTEM_DIR@

SYSTEM_LANG_DIR@

TEMPLATE_CHART_DIR@

Returns the Presents chart template directory

Format TEMPLATE_CHART_DIR@()

Description Returns the pathname of the directory containing the chart templates used by Applixware Presents. By default, this macro returns the directory /*install_dir*/template/chart, where *install_dir* is your Applix*ware* installation directory.

TEXT_CLASSIFY@

Returns values depending on the case of text

Format value = TEXT_CLASSIFY@(text)

Arguments text The text you are checking for case.

Description Classifies text according to the case of the letters contained within the text. One of the following four values will be returned:

0	' AaAa
1	' aaaa
2	' Aaaa
3	' AAAA

See also TEXT_FIX_CASE@

TEXT_DICT_ADD_WORD@

Add a word to the indicated dictionary

Format TEXT_DICT_ADD_WORD@(word, correct, language, dict)

Arguments

word	The word you are adding to the dictionary.
correct	The corrected word you are changing word to.
language	An integer language code as follows:

English	1
German	2

French	3
Spanish	4
Italian	5
British	6
Swedish	7
Danish	8
Norwegian	9
Dutch	10
Portuguese	11
Brazilian	12
Canadian	13
Swiss-German	14
Nynorsk	15
Finnish	16

dict The dictionary to which you are adding the word.

Description Adds a word (or a misspell/correction pair) to a dictionary. If word is illegal, TEXT_DICT_ADD_WORD@ displays an error and continues. If correct is not empty or NULL, word is replaced by correct.

Note that if the corrected word is not phonetically equal to the misspelled word, the word that you added will not be returned.

See also TEXT_DICT_CAN_ADD@

TEXT_DICT_CLOSE@

TEXT_DICT_CREATE@

TEXT_DICT_GET_LANGUAGE@

TEXT_DICT_GET_WORDS@

TEXT_DICT_REMOVE_WORD@

TEXT_DICT_CAN_ADD@

Determines if a dictionary can be added

Format flag = TEXT_DICT_CAN_ADD@()

Description Returns TRUE if another dictionary can be added; otherwise, FALSE is returned. The maximum number of dictionaries is 8.

See also TEXT_DICT_ADD_WORD@

TEXT_DICT_CLOSE@

TEXT_DICT_CREATE@

TEXT_DICT_GET_LANGUAGE@
TEXT_DICT_GET_WORDS@
TEXT_DICT_REMOVE_WORD@

TEXT_DICT_CLOSE@

Closes a dictionary

Format TEXT_DICT_CLOSE@(dictionary)

Arguments dictionary The dictionary you want to close.

See also TEXT_DICT_ADD_WORD@
TEXT_DICT_CAN_ADD@
TEXT_DICT_CREATE@
TEXT_DICT_GET_LANGUAGE@
TEXT_DICT_GET_WORDS@
TEXT_DICT_REMOVE_WORD@

TEXT_DICT_CREATE@

Creates a new, empty dictionary

Format TEXT_DICT_CREATE@(language, dictionary)

Arguments language An integer language code representing the language of the dictionary:

English	1
German	2
French	3
Spanish	4
Italian	5
British	6
Swedish	7
Danish	8
Norwegian	9
Dutch	10
Portuguese	11
Brazilian	12
Canadian	13

Swiss-German	14
Nynorsk	15
Finnish	16

dictionary The dictionary you want to create.

See also TEXT_DICT_ADD_WORD@
TEXT_DICT_CAN_ADD@
TEXT_DICT_CLOSE@
TEXT_DICT_GET_LANGUAGE@
TEXT_DICT_GET_WORDS@
TEXT_DICT_REMOVE_WORD@

TEXT_DICT_GET_LANGUAGE@

Returns the language of a dictionary

Format langCode = TEXT_DICT_GET_LANGUAGE@ (dictionary)

Arguments dictionary The name of a dictionary.

Description Returns the integer code for a language. This code is one of the following values:

English	1
German	2
French	3
Spanish	4
Italian	5
British	6
Swedish	7
Danish	8
Norwegian	9
Dutch	10
Portuguese	11
Brazilian	12
Canadian	13
Swiss-German	14
Nynorsk	15
Finnish	16

If the dictionary file does not exist or has been corrupted, the value 0 (zero) is returned.

See also TEXT_DICT_ADD_WORD@
TEXT_DICT_CAN_ADD@

TEXT_DICT_CLOSE@
TEXT_DICT_CREATE@
TEXT_DICT_GET_WORDS@
TEXT_DICT_REMOVE_WORD@

TEXT_DICT_GET_WORDS@

Retrieve some or all of the words from a dictionary

Format wordArray = TEXT_DICT_GET_WORDS@(language, dictionary, firstTimeFlag, maxGet)

Arguments

language — The integer language code:

English	1
German	2
French	3
Spanish	4
Italian	5
British	6
Swedish	7
Danish	8
Norwegian	9
Dutch	10
Portuguese	11
Brazilian	12
Canadian	13
Swiss-German	14
Nynorsk	15
Finnish	16

dictionary — The dictionary you want to use.

firstTimeFlag — A Boolean value which is set to TRUE the first time you call this function; subsequent uses should set this to FALSE if you are obtaining more words.

maxGet — The maximum number of words to be retrieved.

Description Get some or all of the words from dictionary. firstTimeFlag should be TRUE when you first retrieve words, FALSE for remaining times you retrieve words.

The remaining elements in the wordArray are the returned words.

See also TEXT_DICT_ADD_WORD@

TEXT_DICT_CAN_ADD@
TEXT_DICT_CLOSE@
TEXT_DICT_CREATE@
TEXT_DICT_GET_LANGUAGE@
TEXT_DICT_REMOVE_WORD@

TEXT_DICT_REMOVE_WORD@

Removes a word from a dictionary

Format TEXT_DICT_REMOVE_WORD@(word, correct, language, dictionary)

Arguments

word — The word you want to delete from the dictionary.

correct — The string that will replace word.

language — The integer language code:

English	1
German	2
French	3
Spanish	4
Italian	5
British	6
Swedish	7
Danish	8
Norwegian	9
Dutch	10
Portuguese	11
Brazilian	12
Canadian	13
Swiss-German	14
Nynorsk	15
Finnish	16

dictionary — The dictionary you want to use.

Description Removes a word (or a misspelled/correction pair) from a dictionary. If correct is not null or empty, word is misspelled and is replaced by correct.

See also TEXT_DICT_ADD_WORD@
TEXT_DICT_CAN_ADD@
TEXT_DICT_CLOSE@

TEXT_DICT_CREATE@
TEXT_DICT_GET_LANGUAGE@
TEXT_DICT_GET_WORDS@

TEXT_FIX_CASE@

Adjusts the capitalization of the passed text

Format newText = TEXT_FIX_CASE@(text, template)

Arguments	text	The text whose capitalization is being changed.
	template	A text string that contains letters in the case you want text to be changed. For example, if template contains a lower case string, text will be transformed to lower case. template can be lower, upper, and mixed case.

Description Adjusts the capitalization of the passed text. If template is not of mixed case, TEXT_FIX_CASE@ returns the equivalent version of the passed text.

See also TEXT_CLASSIFY@

TEXT_GET_CURRENT_DICTS@

Returns an array of active dictionaries

Format array = TEXT_GET_CURRENT_DICTS@()

Description Returns an array of dictionary names that are active for the current document.

See also TEXT_SET_CURRENT_DICTS@

TEXT_GET_LANGUAGE_ABBREV@

Returns an abbreviation for a language code

Format abbrev =TEXT_GET_LANGUAGE_ABBREV@(code)

Arguments	code	A three-letter code for one of sixteen languages.

Description Returns a three-letter abbreviation for the passed language code. code is one of the following:

1	English	eng
2	German	grm
3	French	frn
4	Spanish	spn
5	Italian	itl
6	British	brt
7	Swedish	swd
8	Danish	dan
9	Norwegian	nrw
10	Dutch	dut
11	Portuguese	prt
12	Brazilian	brz
13	Canadian	cfr
14	Swiss-German	sgr
15	Nynorsk	nyn
16	Finnish	fin

See also TEXT_GET_LANGUAGE_CODE@

TEXT_GET_LANGUAGE_CODE@

Returns an integer code for a language

Format code = TEXT_GET_LANGUAGE_CODE@ (languageName)

Arguments languageName
The name of one of the sixteen languages available with Applix*ware*.

Description Returns an integer code for the language. Note most other speller functions will need this code. The constant NOLANG (0) is returned if the string is not recognized. The fact that the spell checker recognizes the language name does not mean it is supported. code is one of the following:

English	1
German	2
French	3
Spanish	4
Italian	5
British	6
Swedish	7
Danish	8
Norwegian	9
Dutch	10
Portuguese	11

Brazilian	12
Canadian	13
Swiss-German	14
Nynorsk	15
Finnish	16

See also TEXT_GET_LANGUAGE_ABBREV@
TEXT_GET_LANGUAGE_NAME@

TEXT_GET_LANGUAGE_NAME@

Returns the language name corresponding to a language code

Format langName = TEXT_GET_LANGUAGE_NAME@ (langCode)

Arguments langCode An integer language code.

Description Returns the name of the language that corresponds to a language code. An empty string is returned if the integer is not in the following range:

0 < langCode < 17

See TEXT_GET_LANGUAGE_CODE@ for a table of language codes and the language names associated with these codes.

TEXT_GET_USER_DICT_NAME@

Returns the default dictionary path name

Format pathname = TEXT_GET_USER_DICT_NAME@(langCode)

Arguments langCode The integer language code:

English	1
German	2
French	3
Spanish	4
Italian	5
British	6
Swedish	7
Danish	8
Norwegian	9
Dutch	10
Portuguese	11

Brazilian	12
Canadian	13
Swiss-German	14
Nynorsk	15
Finnish	16

Description Returns the path name of the default user dictionary used for a particular language.

TEXT_HYPHENATE@

Returns hyphenation information array

Format hyphArray = TEXT_HYPHENATE@(word, language, dictionaryArray, rankFlag, how)

Arguments

word — The text for which you are determining hyphenation points.

language — One of the following integer language codes:

English	1
German	2
French	3
Spanish	4
Italian	5
British	6
Swedish	7
Danish	8
Norwegian	9
Dutch	10
Portuguese	11
Brazilian	12
Canadian	13
Swiss-German	14
Nynorsk	15
Finnish	16

dictionaryArray — A null-terminated array of dictionaries.

rankFlag — A Boolean indicating whether hyphenation points are ranked.

how — The way to generate hyphenation points: HYPH_NONE, HYPH_CALC, or HYPH_DICT.

Description Hyphenation information is returned in a single array as follows:

0 How the hyphenation points were generated (a HYPH_XXXX value).

1	Hyphenation points for the word: an array of integers, one for each character in the word, describing hyphenation point following the character (if any).

TEXT_SET_CURRENT_DICTS@

Returns an array of active dictionaries

Format TEXT_SET_CURRENT_DICTS@(arrayOfDicts)

Arguments

arrayOfDicts	an array of absolute pathnames to Applixware dictionary files. Dictionary files must have a .dct extension.

Description Establishes a set of dictionaries for the current document.

See also TEXT_GET_CURRENT_DICTS@

TEXT_SIZE@

Returns the size of 'text'

Format sizeArray = TEXT_SIZE@(text, len[, font, pointSize[, weight[, slant]]])

Arguments

text	The text string.
len	The number of characters in the text.
font	The font used to display the text. If this value is NULL, the Applix*ware* default font will be used.
pointSize	The point size
weight	A number indicating if weight is added to the character, as follows: 0 non-bold; 1 bold
slant	A number indicating if the text is printed at an angle, as follows: 0 non-italic; 1 italic

Description Returns an array containing:

- the width (sizeArray[0])
- height (sizeArray[1])
- the ascent (sizeArray[2])

The values returned are in screen pixels. While the conversion between pixels and points is dependent upon your display resolution, a typical conversion factor is 40/3.

TEXT_SPELLED_CORRECTLY@

Determines if a word is spelled correctly

Format flag =TEXT_SPELLED_CORRECTLY@(word, language, dictionaryArray)

Arguments

word — The word you want to check for correct spelling.

language — The integer language code

English	1
German	2
French	3
Spanish	4
Italian	5
British	6
Swedish	7
Danish	8
Norwegian	9
Dutch	10
Portuguese	11
Brazilian	12
Canadian	13
Swiss-German	14
Nynorsk	15
Finnish	16

dictionaryArray
An array of dictionary names; this list is a null-terminated array.

Description Determines if the word is spelled correctly and returns TRUE if it is. Otherwise, FALSE is returned.

If word exceeds 31 characters, TEXT_SPELLED_CORRECTLY@ returns the following error:

word is too long for spelling

See also TEXT_SPELLER_CLOSE@

TEXT_SPELLER_CURR_LANGUAGE@

TEXT_SPELLER_HYPHENATION_OK@

TEXT_SPELLER_LANGUAGE_OK@

TEXT_SPELLER_THESAURUS_OK@
TEXT_SPELLING_ALTERNATIVES@
TEXT_SPELLING_DATA@
TEXT_SPELLING_ERRORS@
TEXT_THESAURUS@

TEXT_SPELLER_CLOSE@

Closes open spelling system

Format TEXT_SPELLER_CLOSE@( )

Description Completely closes the Applix*ware* spell checker. This macro closes the thesaurus, hyphenation processor, and spelling dictionaries, as well as removing the language environment.

See also TEXT_SPELLED_CORRECTLY@
TEXT_THESAURUS@
TEXT_THESAURUS_CLOSE@

TEXT_SPELLER_CURR_LANGUAGE@

Returns current language

Format langCode = TEXT_SPELLER_CURR_LANGUAGE@()

Description Returns the current language as an integer between 1 and 16. The following lists the available languages:

English	1
German	2
French	3
Spanish	4
Italian	5
British	6
Swedish	7
Danish	8
Norwegian	9
Dutch	10
Portuguese	11

Brazilian	12
Canadian	13
Swiss-German	14
Nynorsk	15
Finnish	16

See also TEXT_SPELLED_CORRECTLY@
TEXT_SPELLING_ALTERNATIVES@

TEXT_SPELLER_HYPHENATION_OK@

Checks to see if hyphenation dictionary files are available

Format flag = TEXT_SPELLER_HYPHENATION_OK@ (langCode)

Arguments langCode An integer representing the languages:

English	1
German	2
French	3
Spanish	4
Italian	5
British	6
Swedish	7
Danish	8
Norwegian	9
Dutch	10
Portuguese	11
Brazilian	12
Canadian	13
Swiss-German	14
Nynorsk	15
Finnish	16

See also TEXT_SPELLED_CORRECTLY@
TEXT_SPELLER_LANGUAGE_OK@
TEXT_SPELLER_THESAURUS_OK@

TEXT_SPELLER_LANGUAGE_OK@

Checks to see if spelling dictionaries are available

Format flag =TEXT_SPELLER_LANGUAGE_OK@(langCode)

Arguments langCode An integer representing one of the following languages:

English	1
German	2
French	3
Spanish	4
Italian	5
British	6
Swedish	7
Danish	8
Norwegian	9
Dutch	10
Portuguese	11
Brazilian	12
Canadian	13
Swiss-German	14
Nynorsk	15
Finnish	16

See also TEXT_SPELLED_CORRECTLY@
TEXT_SPELLER_HYPHENATION_OK@
TEXT_SPELLER_THESAURUS_OK@

TEXT_SPELLER_THESAURUS_OK@

Checks to see if thesauri files are available

Format flag = TEXT_SPELLER_THESAURUS_OK@(langCode)

Arguments langCode An integer representing one of the following languages:

English	1
German	2
French	3
Spanish	4
Italian	5

British	6
Swedish	7
Danish	8
Norwegian	9
Dutch	10
Portuguese	11
Brazilian	12
Canadian	13
Swiss-German	14
Nynorsk	15
Finnish	16

See also TEXT_SPELLED_CORRECTLY@
TEXT_SPELLER_HYPHENATION_OK@
TEXT_SPELLER_LANGUAGE_OK@

TEXT_SPELLING_ALTERNATIVES@

Returns a list of suggested spellings

Format wordArray =TEXT_SPELLING_ALTERNATIVES@(word, languages, dictionaryArray, maxGet)

Arguments

word The word being checked.

language One of the following integer language codes:

English	1
German	2
French	3
Spanish	4
Italian	5
British	6
Swedish	7
Danish	8
Norwegian	9
Dutch	10
Portuguese	11
Brazilian	12
Canadian	13
Swiss-German	14
Nynorsk	15
Finnish	16

dictionaryArray	A null-terminated array of dictionary names.
maxGet	The maximum number of words desired.

Description Returns an array of words that are alternatives to the way that word is spelled. The array of returned information is organized as follows:

0:	The number of meanings returned.
1:	TRUE if word is in the list; FALSE otherwise.
2:	Array of spelling alternatives.

See also TEXT_SPELLED_CORRECTLY@
TEXT_SPELLER_CURR_LANGUAGE@
TEXT_SPELLING_DATA@
TEXT_SPELLING_ERRORS@

TEXT_SPELLING_DATA@

Returns an array of dictionary names

Format dictionaryArray = TEXT_SPELLING_DATA@()

Description Returns a null-terminated list of dictionary names

See also TEXT_SPELLED_CORRECTLY@
TEXT_SPELLING_ALTERNATIVES@
TEXT_SPELLING_ERRORS@

TEXT_SPELLING_ERRORS@

Returns information about the first spelling error found in the passed string

Format errorArray = TEXT_SPELLING_ERRORS@(string, startOff, language, dictionaryArray, misspellings)

Arguments

string	A string being examined for errors
startOff	The offset within string at which spell checking will begin.
language	The language code:
	English 1

German	2
French	3
Spanish	4
Italian	5
British	6
Swedish	7
Danish	8
Norwegian	9
Dutch	10
Portuguese	11
Brazilian	12
Canadian	13
Swiss-German	14
Nynorsk	15
Finnish	16

dictionaryArray
A null-terminated array of dictionaries.

misspellings An array of speller codes indicating the conditions to be checked. If this array is NULL, the profile will be used.

Description Spelling error information is returned in a single array as follows:

0	No errors found.
1	Starting character offset of error .
2	Ending character offset of error.
3	The last character the spell checker examined to generate this error.

See also TEXT_SPELLED_CORRECTLY@

TEXT_SPELLER_CURR_LANGUAGE@

TEXT_SPELLING_ALTERNATIVES@

TEXT_SPELLING_DATA@

TEXT_THESAURUS@

Returns thesaurus information

Format infoArray = TEXT_THESAURUS@(word, language)

Arguments word The word you are looking up in the thesaurus.

language The language code:

English	1
German	2

French	3
Spanish	4
Italian	5
British	6
Swedish	7
Danish	8
Norwegian	9
Dutch	10
Portuguese	11
Brazilian	12
Canadian	13
Swiss-German	14
Nynorsk	15
Finnish	16

Description Thesaurus information is returned in a single array as follows:

0: Number of meanings returned
1: Data for first meaning, in the following array:
   a: First meaning string
   b: List (i.e. subarray) of synonyms for 1st meaning
   c: List of compared words for 1st meaning
   d: List of related words for 1st meaning
   e: List of contrasted words for 1st meaning
   f: List of antonyms for 1st meaning
2: Data for second meaning, in the following array: (and so on)

See also TEXT_SPELLED_CORRECTLY@
TEXT_SPELLER_CURR_LANGUAGE@
TEXT_SPELLER_THESAURUS_OK@
TEXT_THESAURUS_CLOSE@

TEXT_THESAURUS_CLOSE@

Closes the thesaurus file

Format TEXT_THESAURUS_CLOSE@()

Arguments None.

Description Closes the thesaurus file. If a problem occurs, ELF throws an error.

See also TEXT_SPELLER_CLOSE@

TEXT_THESAURUS@

TIME@

Converts a complete time into a serial time number

Format	TIME@(hour, minute, second)

Arguments	hour	A number between 0 and 23.
	minute	A number between 0 and 59.
	second	A number between 0 and 59.

Description The TIME@ function converts a complete time (hours, minutes, and seconds) into a time number, which is a decimal between 0 and 1. For example:

```
TIME@(0,0,1)   returns 0.000011574
TIME@(23,59,59)  returns 0.999988426
TIME@(12,0,0)   returns 0.5
```

Applix*ware* checks the time you enter to make sure it is a valid entry. For example, if you enter TIME@(24,12,37), your macro throws an error because valid hour arguments include the whole numbers from 0 to 23 and do not include the number 24.

TIME_SLICE@

Suspends the current task after a specified period

Format	TIME_SLICE@(milliSeconds)

Arguments	milliSeconds	The number of seconds allocated to the current task. After this period has elapsed, the current task is suspended.

Description The TIME_SLICE@ macro works with the @@@ TIME_SLICE and @@@ NO_TIME_SLICE pragmas. The purpose of TIME_SLICE@ is to allocate a period of time for execution of the current macro. After this period has elapsed, the current task is suspended, and the ELF Scheduler can allocate time to other ELF tasks in its queue.

Between the time when the TIME_SLICE@ macro is processed and the expiration of the specified time period, ELF checks to see how much time has elapsed *after every line of your ELF macro*. This adversely affects response time. For this reason, the TIME_SLICE@ macro should be used very carefully, and only in small segments of

code that are delimited by the
@@@ TIME_SLICE and @@@ NO_TIME_SLICE pragmas.

```
macro slice

var x
NEW_TASK_UNPENDED@("test")          ' Schedule the test Macro, but maintain
                                    ' the execution thread.

@@@ TIME_SLICE                      ' Enable time checking after each line
TIME_SLICE@(5000)                   ' Set a 5 second timeout

x = 0
    while x <> 100000               ' Set this to 100 to avoid the timeout.
            x = x+1
    wend

TIME_SLICE@(0)                      ' Disable the timeout
@@@ NO_TIME_SLICE                   ' Disable time checking

info_message@("This runs first if the timeout does not occur!")
endmacro


macro test

info_message@("Timed out!")         ' This Macro runs if the 5-second time limit expires

endmacro
```

See also ELF Scheduler, Tasks

TIMESTR@

Returns a formatted time string from a Time Value number

Format TIMESTR@(timeNumber, format)

Arguments

timeNumber	a decimal between 0 and 1 as returned by the macro TIMEVALUE@.
format	a number indicating the format of the string returned by TIMESTR@.

Description TIMESTR@ returns a formatted time string. The timeNumber is a time value number. It is a decimal value between 0 and 1. This value is returned by the macro TIME-VALUE@. The following shows a few examples of time value numbers:

TIMEVALUE@("12:00:01 am") returns 0.0000116
TIMEVALUE@("11:59:59 pm") returns 0.999988426
TIMEVALUE@("12:00:00 pm") returns 0.5

format is a number between 0 and 6. This argument determines the format of the string. Several examples follow:

Example	**Returned String**
TIMESTR@(.5, 0)	12:00:00 PM
TIMESTR@(.5, 1)	12:00 PM
TIMESTR@(.5, 2)	12:00:00
TIMESTR@(.5, 3)	12:00
TIMESTR@(.5, 4)	12:00:00.000 PM
TIMESTR@(.5, 5)	12:00:00.000
TIMESTR@(.5, 6)	12:00:00

If format is greater than 6, TIMESTR@ returns the same value as TIMESTR@(*n*, 6).

See also TIMEVALUE@

TIMEVALUE@

Converts a formatted Applixware time string into a time number

Format TIMEVALUE@(timeString)

Arguments timeString   a formatted time string

Description Converts a formatted Applixware time string into an Applixware time number. A time number is a decimal between 0 and 1. For example:

TIMEVALUE@("12:00:01 am") returns 0.0000116
TIMEVALUE@("11:59:59 pm") returns 0.999988426
TIMEVALUE@("12:00:00 pm") returns 0.5

The file datetim_.am contains valid formats for timeString.

See also Time Formats in the Applixware Words online help lists some default time formats.

TODAY@

Returns the current date as a serial number

Format TODAY@( )

Description The TODAY@ function returns the current date as a serial number. For example, if today's date is August 6, 1984, TODAY@ returns 30899. This function does not require an argument; however, it must be followed by open and closed parentheses.

TRIM@

Returns a string that has no leading or trailing spaces

Format newString = TRIM@(string)

Arguments	string	A text string that may have leading and/or trailing spaces.

Description Removes leading and trailing spaces from text.

Example

TRUNC_ARRAY@

Truncates an array

Format newArray = TRUNC_ARRAY@(array, newSize)

Arguments	array	The ELF array you want to truncate. array may contain strings, numbers, or arrays.
	newSize	A number indicating the size (number of elements) to make array. newSize must be greater than or equal to 0.

Description Reduces the size of an array. If the array size specified by newSize is greater than or equal to the size of the existing array, then only the number of elements in the existing array are returned.

See also SUBARRAY@

TYPE@

Types a string of characters

Format TYPE@(text)

Arguments The string of characters to type.

Description Enters characters into the current Applix*ware* window at the current cursor position. In Spreadsheets windows, the characters are typed in the entry area. After this edit, the cursor is to the right of the text just entered.

TYPE@ cannot be used for typing text in dialog boxes. TYPE@ does not recognize the newline (\n) character.

Example

UNBIND_C_LIBRARY@

Detaches C shared library; auto-rebind can still occur

Format UNBIND_C_LIBRARY@(pathname)

Arguments	pathname	The path name of a shared library previously bound into the Applix*ware* environment.

Description Unbinds the shared library from the Applix*ware* environment. This action is not usually performed. Instead, this macro exists as a convenience to developers to that they can change a shared library without bringing Applix*ware* down.

Although a library may be unbound, you do not need to re-execute **INSTALL_C_LIBRARY@** again to access the functions in a newer version of a library. Instead, the library is automatically rebound when any of the functions within it are used.

Executing this macro resets static data.

UNIX_PROCESS_ID@

Returns the UNIX process id (PID) number

Format pid = UNIX_PROCESS_ID@()

Example

UNIX_USER_ID@

Returns the UNIX User ID

Format UNIX_USER_ID()

UNQUOTE_STRING@

Removes double quotes from a passed string

Format flag = UNQUOTE_STRING@(string)

Arguments string The name of the string being passed.

Description If string is bounded by double quotes, UNQUOTE_STRING@ removes the quotes and returns TRUE. Otherwise, string is left alone and FALSE is returned. Quotes within string are ignored. Also, if string only has one quote, it is ignored and FALSE is returned (with no change to string).

UNREGISTER_OBJECT@

Removes an object from the Applixware registry

Format UNREGISTER_OBJECT@(name)

Arguments name The registered name of an object in the Applixware registry

Description Removes an object from the Applixware registry. This object must have been previously registered with the function **REGISTER_OBJECT@**.

The Applix*ware* object registry is maintained in the file bldrObjects.reg file in your ax-home directory.

UPDATE_BITMAP@

Re-loads a bitmap into memory

Format UPDATE_BITMAP@(filename)

Arguments	filename	The name of the bitmap. This name does not include the .im extension.

Description Re-loads a given bitmap into memory. This macro is necessary only to view any edits made to a bitmap (during this session) outside of the Applixware pixel editor.

UPDATE_LINKS_INFO_FROM_LIST@

Updates link information

Format UPDATE_LINKS_INFO_FROM_LIST@(format doc_links_info@ info, format arrayof link_info@ linksList)

Arguments	info	An array of link information. This information is typically created using GET_LINKS_INFO@.
	linksList	A list of link information. This list is typically created using GET_LINKS_LIST@.

Description Updates a document's link information based upon information in linksList.

UPDATE_SELECTIONS_FILE@

Updates a menu bar definition and updates the associated disk file

Format UPDATE_SELECTIONS_FILE@(id, file, array)

Arguments id

Indicates which type of menu bar is being updated. For custom (non-default) menu bars, choose from one of the following ranges:

0 to 20	reserved--do not use.
21 to 99	custom dialog boxes.
100 to 199	custom Spreadsheets menu bars.
200 to 299	custom Words menu bars.
300 to 399	custom Graphics menu bars.
400 to 499	custom Macro Editor menu bars.

If instead you want to modify a default menu bar, choose one of these ids:

0	Dialog box
1	Words
2	Graphics
3	Spreadsheets
4	Macro Editor
5	Main Menu

	6	Directory Displayer
	11	Applixware Inbox

file — The path name to which you are writing the ELF data file containing the menu bar's definition. If only the file name is specified, the path defaults to the user's home directory. file need not coincide with a ``default'' Applix*ware* menu bar.

array — Name of the array containing the menu bar definition. If the id is less than 20, the array must coincide with the corresponding default ax file (either ax_ss4, ax_wp4, ax_gr4, ax_me4, ax_dd4, ax_mm4, ax_inb4, or ax_dlg4).

Description If id is an application window, all open windows using the file menu bar or a default menu bar (as listed below) will automatically and instantly display the newly updated array menu bar.

When customizing an application window's menu bar, you may or may not want to save the changes to the ``default'' menu bar. By default, Applix*ware* loads into memory the menu bar definition contained in one of the following ELF data files:

ax_ss4	Spreadsheets
ax_wp4	Words
ax_gr4	Graphics
ax_me4	Macro Editor

NOTE: If you load a custom menu bar into an application window, you cannot further modify the menu bar by choosing Customize Menu Bar. Customize Menu Bar accesses only the default menu bar for the given application.

See also SET_SELECTIONS@

SET_MENU_BAR_ID@

UPPERCASE@

Returns the ASCII uppercase version of a string

Format upText = UPPERCASE@(text)

Arguments text — A string that contains any combination of uppercase and lowercase characters.

Description Strings in ELF are case sensitive. This means that text strings must be converted to the same case if you want to compare them.

See also LOWERCASE@

USE_WIDE_MENUBAR_EDIT@

Indicates if wide fields are used

Format flag = USE_WIDE_MENUBAR_EDIT@( )

Description Returns a Boolean value indicating if wide fields should be used when editing menu bars.

USER_DIR@

Returns the name of the users' home directory

Format userDir = USER_DIR@()

Description Returns the name of the users' home directory as defined by the $HOME environment variable.

Example

See also AXHOME_DIR@
CURRENT_DIR@

USER_NAME@

Returns the user name associated with the current Applix*ware* window

Format userName = USER_NAME@()

VALID_FILE_NAME@

Determines if a name is a valid file name

Format flag = VALID_FILE_NAME@(name)

Arguments	name	A string containing the file name whose name is being checked for validity.

Description Determines if name follows the requirement that a name be composed of any combination of the characters: a - z, A - Z, 0 - 9, back slash (/), period (.), underscore (_). VALID_FILE_NAME@ returns TRUE if name is valid and FALSE if name is invalid.

See also VALID_MACRO_NAME@

VALID_MACRO_NAME@

Indicates whether a name conforms to ELF naming conventions

Format VALID_MACRO_NAME@(name)

Arguments A string containing the name of the macro whose characters are being checked.

Description Throws an error if the macro specified by name is not a valid macro. If the specified macro is valid, processing continues with the statement following VALID_MACRO_-NAME@.

See also VALID_FILE_NAME@

VERIFY_FILE_EXISTS@

Checks to see if a file exists

Format VERIFY_FILE_EXISTS@(name)

Arguments

name	The name of the file whose existence you want to verify.

Description Determines if a file exists. If it does not, this macro throws an error.

VERIFY_MENU@

Verifies a menus structure

Format VERIFY_MENU@(menuArray,menuName)

Arguments

menuArray	The array of menu information.
menuName	The menu name.

Description Performs the following checks:

· Checks for valid accelerator syntax.

· Checks for non-duplicate accelerators.

· Checks for non-duplicate mnemonics.

If a problem occurs, an error is thrown.

VERIFY_NAME@

Verifies whether a valid function name is being used

Format errorCode = VERIFY_NAME@(name)

Arguments name The name of the function you want to verify.

Description Returns an error if name is not valid. Possible errorCodes are:

ERR_BADNAME	Illegal characters
ERR_KEYWORD	Name is a keyword
ERR_MYNAME	Collides with an Applixware name
ERR_RSVDWORD	Name is a reserved word
ERR_STRTOOLONG	String is too long

VERSION@

Returns the Applix*ware* version

Format version = VERSION@()

Description Returns the Applix*ware* version number. (This is actually a time stamp indicating when it was built.)

VERSION_STRING@

Returns the Applix*ware* version number

Format version = VERSION_STRING@()

Description Returns a string that indicates the version number of the Appplix*ware* currently being run. For example, it could return the following value:

4.2 (0)

VIEW_ONLY_LICENSE@

Indicates if you are using a view-only license

Format flag = VIEW_ONLY_LICENSE@()

Description Returns TRUE if you are running Applix*ware* using a view-only license.

WAIT_WHILE_BUSY@

Causes ELF to delay the start of a new operation until a current operation is completed

Format WAIT_WHILE_BUSY@()

Description Prevents ELF from performing an action before it completes an operation that requires a period of time to execute. For example, if your ELF macro opens multiple windows and types text into the windows, this macro could be used to delay typing until the process of opening an application window is completed.

It should not be necessary to use WAIT_WHILE_BUSY@. Try running commands without the use of WAIT_WHILE_BUSY@. If the functioning of a macro appears to be affected by the amount of time operations take to complete, use WAIT_WHILE_BUSY@ to delay the start of operations.

WEEKDAY@

Returns the days of the week for a date value

Format WEEKDAY@(dateValue, returnType)

Arguments

dateValue	A number representing the date.
sequenceNumber	a number from 1 to 3 indicating which numbering sequence to use.

Description WEEKDAY@ returns the day of the week corresponding to the date value. The date value is the number of calendar days after 12/31/1899 that the given date falls. This number is returned by the macro DATEVALUE@. For example, DATEVALUE@("1/1/1900") is 1, and DATEVALUE@("12/31/1995") is 35,430.

For sequenceNumber, you can define the numbering sequence of the days of the week:

Return_type	Day number returned
1 or omitted	Numbers 1 (Sunday) through 7 (Saturday).
2	Numbers 1 (Monday) through 7 (Sunday).
3	Numbers 0 (Monday) through 6 (Sunday).

The returned value is a number from 0 to 7.

WILDCARDER@

Returns either array index of match or strings that match

Format array = WILDCARDER@(wildSpec, strings, returnFlag)

Arguments

wildSpec	The wildcard specification. Wildcards that can be used are:
	* Matches any string of characters. This string includes the NULL string.
	? Matches any single character.
	[abc] Matches any of the enclosed characters.
	[!abc] Does not match any of the enclosed characters.
strings	An array of strings to be searched.
returnFlag	A Boolean value which if TRUE indicates that the returned array contains numbers where each number is the zero-based position indicating that the text within array element matched the wildcard specification. If this value is FALSE, the returned array contains the strings that match wildSpec.

Description Returns an array whose contents either contain the array element within string that had a match or the substring index position at which the match occurred.

For example, assume you have the following array declaration:

foob = { "athe", "kjlk", "other", "them", "those", "ldfdf" }

The value returned from:

WILDCARDER@("*t*e*", foob, TRUE)

is:

{ 0, 2, 3, 4 }

This indicates that those four array elements contain text that matches the wildcard specification. The following example shows how you can create an array that parallels the original array's structure:

```
macro foo
    var foob, i, hits, ls

    foob = { "athe", "kjlk", "other", "them", "those", "ldfdf" }
    ls = wildcarder@("*t*e*", foob, TRUE)

    for i = 0 to array_size@(ls) -1
        hits[ls[i]] = TRUE
    next i

    dump_array@(hits)
endmacro
```

WINDOW_INFO@

Displays information about an Applix*ware* window

Format var format window_format@ = WINDOW_INFO@(number)

Arguments The small or large identification number of an Applix*ware* window.

Description Returns the following information about the specified window. The FORMAT template name is window_format@. The header file containing the FORMAT template is windows_.am.

The definition of window_format@ is as follows:

format window_format@

id, The small window id, which is a number from 0 to 21.

title, A string indicating the window's title of the window.

icon_id,
The set-aside icon used for the window. The possible icon_id values are:

Definition	Num.	Description
APP#WORDS_	1	Words
APP#GRAPHICS_	2	Graphics
APP#SPREAD-SHEETS_	3	Spread-sheets
APP#MACRO_-EDIT_	4	Macro Edi-tor
APP#MAIN_-MENU_	5	Main Menu

APP#DIR_DISP_	6	Directory Displayer
APP#DLG_EDIT_	8	(old) Dialog Box editor
APP#BIT_EDIT_	9	Bitmap Editor
APPP#QUERY_	10	Data
APP#EQUATION_	12	Equation Editor
APP#SENDMAIL_	17	Send Mail
APP#INBOX_	19	Inbox
APP#DLGED_	20	Dialog Editor
APP#ARCADE	23	Builder

task_id,
The task number that owns this window.

menubar_id,
The window's menu bar id.

The icon_id application type definitions are defined in the app_ids_.am header file. You must use the INCLUDE "app_ids_.am" statement at the start of your macro document to use these definitions.

Example

See also LIST_OF_WINDOWS@
SELECT_WINDOW@

WINDOW_LARGE_ID@

Returns the large id of an Applix*ware* window

Format var id = WINDOW_LARGE_ID@(number)

Arguments The small identification number of an Applix*ware* window.

Description Returns the corresponding large identification number (a number starting from 100001), or -1 if the small identification number does not exist.

WINDOW_SMALL_ID@

Returns the small id of an Applix*ware* window

Format var id = WINDOW_SMALL_ID@(number)

Arguments The large identification number of an Applix*ware* window.

Description Returns the corresponding small identification number (a number from 0 to 21), or -1 if the large identification number does not exist.

WORKDAY@

Calculates the date that is a number of days before or after a start date

Format WORKDAY@(startDate, days, holidays)

Arguments

startDate	The date value of the day from which to start the calculation.
days	The number of days before or after the start date
holidays	holidays to be excluded from the calculation

Description WORKDAY@ calculates the date value that is a number of whole working days before or after startDate.

The startDate argument is a date value. The date value is the number of calendar days after 12/31/1899 that the given date falls. This number is returned by the macro DAT-EVALUE@. For example, DATEVALUE@("1/1/1900") is 1, and DATEVALUE@("12/31/1995") is 35,430.

The days argument is a number working days, excluding weekends and any days identified as holidays. This is the number of days to add or subtract from startDate to yield the final result.

For holidays, you may enter an optional set of one or more date values representing days to exclude from the working calendar. Several examples follow:

WORKDAY@(DATEVALUE@("01/12/95"), 5)

This macro calculates five working days after 1/12/95. It yields the date value 34718 or 01/19/95.

WORKDAY@(DATEVALUE@("01/12/95"), 5, DATEVALUE@("01/16/95"))

This macro calculates five working days after 1/12/95. January 16, 1995 is treated as a holiday. It yields 34719 or 01/20/95.

See also DATEVALUE@

WORK_IN_PROGRESS@

Posts a dialog box that can be updated while a task is being performed

Format WORK_IN_PROGRESS@(argument[, title])

Arguments

argument	The text to display in the box. If this text is the empty string (that is, ""), the posted box is deleted.
title	An optional title to display in the dialog box.

Description Displays a dialog box containing a message. This message can be updated while a process is running. For example, the following macro posts a new value within a dialog box every 2 seconds. After the loop ends, this message box is deleted:

```
macro show_in_progress
	var i

	for i = 1 to 20
		WORK_IN_PROGRESS@(" Work accomplished: "
				++ 5*i, "Sample Message" )
		delay@(1)
	next i
	delay@(3)
	WORK_IN_PROGRESS@("")
endmacro
```

WPX_ASSIGN_INSTANCE@

Assigns a Words handle to an Applixware Task

Format WPX_ASSIGN_INSTANCE@(wpx, taskid)

Arguments

wpx	An Applixware Words handle as returned by WPX_CREATE_INSTANCE@.
taskId	An Applix*ware* task ID number.

Description Assigns the Words handle indicated by wpx to task taskid. This macro changes the task's owner from its current task to a different task.

The wpx handle changes during the execution of the macro. The original handle is set by the programmer on input. The new handle is returned to the calling program after the successful completion of the call.

WPX_COPY_INSTANCE@

Creates a Copy of a Words handle

Format WPX_COPY_INSTANCE@(wpx, taskid, newWpx)

Arguments

wpx	An Applixware Words handle as returned by WPX_CREATE_INSTANCE@.
taskId	An Applix*ware* task ID number.
newWpx	A copy of the wpx handle. On input, set this argument to NULL.

Description Creates a copy of the Applixware Words handle wpx. The wpxNew argument becomes an exact copy of the wpx argument.

WPX_CREATE_INSTANCE@

Creates an instance of Applixware Words in memory

Format WPX_CREATE_INSTANCE@(wpx, uid)

Arguments

wpx	The Applixware Words handle to create. On input, set this variable to NULL. After WPX_CREATE_INSTANCE@, this variable contains an Ap-plixware Words handle.
uid	An Applixware task ID. This is an optional value indicating the task to which the Words handle is assigned. If this argument is omitted or set to zero (0), the current task is used.

Description Creates an internal instance of the Applixware Word Processor. A *handle* to this internal editor is returned. You can associate an Applixware Words file with this handle (WP_READ_FILE@) and display the Applixware Words file as an inset in a dialog box (DB_CTRL_INSET@).

WPX_DESTROY_INSTANCE@

Frees resources associated with an Applixware Words handle

Format	WPX_DESTROY_INSTANCE@(wpx)	
Arguments	wpx	The Applixware Words handle to destroy.
Description	Frees all the resources associated with a wpx handle. After running this macro, you can no longer use the Words handle to manipulate the Words object.	

WPX_GET_INSTANCE@

Returns a task's Words handle

Format	WPX_GET_INSTANCE@(wpx, taskID)	
Arguments	wpx	The Applixware Words handle to create. On input, set this variable to NULL.
	uid	An Applixware task ID. This is an optional value indicating the task to which the Words handle is assigned. If this argument is omitted or set to zero (0), the current task is used.
Description	Returns the Words handle associated with an Applixware task.	

WPX_READ_BUFFER@

Associates a data buffer with a Words handle

Format	WPX_READ_BUFFER@(wpx, buffer)	
Arguments	wpx	an Applixware Words handle as returned by WP_CREATE_INSTANCE@.
	buffer	A buffer containing the data from a Words document.
Description	Reads the contents of a buffer and associates the data with a Words handle.	

WPX_READ_FILE@

Reads a Words file and associates the data with a Words handle

Arguments	wpx	A Words handle as returned by the macro WPX_CREATE_INSTANCE@.
	filename	The name of the file being read.

Description Reads the contents of a Words file and assigns it to the Words handle wpx.

While the file is being read, WPX_READ_FILE@ places a read lock on the file. This lock is cleared when the file read is complete. If two ELF macros try to read the same file simultanously, one of the ELF macros will throw an error indicating that the file is locked.

WPX_WRITE_BUFFER@

Writes data associated with a Words handle to a memory buffer

Format WPX_WRITE_BUFFER@(wpx, buffer)

Arguments	wpx	an Applixware Words handle as returned by WP_CREATE_INSTANCE@.
	buffer	An initialized ELF variable

Description Reads the contents of a buffer and associates the data with a Words handle.

WPX_WRITE_FILE@

Reads a Words file and associates the data with a Words handle

Arguments	wpx	A Words handle as returned by the macro WPX_CREATE_INSTANCE@.
	filename	The absolute pathname of a file.

Description Writes the data associated with a Words handle to a file.

While the file is being written, WPX_WRITE_FILE@ places a write lock on the file. This lock is cleared when the file write is complete. If two ELF macros try to write the same file simultanously, one of the ELF macros will throw an error indicating that the file is locked.

WRITE_ASCII_FILE@

Writes an ELF string array to an ASCII file

Format WRITE_ASCII_FILE@(file, array)

Arguments

file	The name of the file (a string) in which you want to store the strings. If you do not specify a path name, the current directory is used. If file is an existing file, the contents of file are overwritten.
array	The name of a single-dimensional string array you want written to file.

Description Opens an operating system ASCII file, writes an array of strings to file, and then closes file. Each string in the array appears on a separate line when written to file.

While the write is in progress, WRITE_ASCII_FILE@ places a write lock on the file. This lock is cleared when the file write is complete. If two ELF macros try to write the same file simultanously, one of the ELF macros will throw an error indicating that the file is locked.

See also READ_ASCII_FILE@.

WRITE_BINARY_FILE@

Writes a binary object to a file

Format WRITE_BINARY_FILE@(file, object)

Arguments

file	The file name to which you want to write the binary object.
object	The binary data object that you want to write to a file.

Description Writes a binary object to a file. Binary files can be read using READ_BINARY_FILE@.

While the write is in progress, WRITE_BINARY_FILE@ places a write lock on the file. This lock is cleared when the file write is complete. If two ELF macros try to write the same file simultanously, one of the ELF macros will throw an error indicating that the file is locked.

See also READ_BINARY_FILE@

WRITE_DATA_FILE@

Writes an ELF array to an ELF data file

Format WRITE_DATA_FILE@(file, array[, format write_data_format@ data])

Arguments

file	The name of the ELF data file in which to write the ELF array. If you do not specify a path name, the current directory is used. If file is the name of an existing file, the contents of file are overwritten.
array	The ELF data you want written to file.
data	Sets the group and all access privileges of the file. The definition of this format is described below:

Description Opens file for writing and writes the ELF array elements to file. WRITE_DATA_FILE@ closes file when writing is completed.

While the write is in progress, WRITE_DATA_FILE@ places a write lock on the file. This lock is cleared when the file write is complete. If two ELF macros try to write the same file simultanously, one of the ELF macros will throw an error indicating that the file is locked.

The definitions for the formats used in this macro are as follows:

```
format write_data_format@
    comments,          'Array of strings to be added to the file as comments
    grp_access,
    all_access,

                       'Specs for header line
    format afile_info file_header
format afile_info
    encoding,          'relative name of doc (suffix suppressed)
    type,              'One of the following:
                       'FSMAT#DIRECTORY
                       'FSMAT#FILE
                       'FSMAT#TEXT_
                       'FSMAT#PICTURE_
                       'FSMAT#SPREADSHEET_
                       'FSMAT#COMMAND_
                       'FSMAT#DIALOG_
                       'FSMAT#BITMAP_
                       'FSMAT#EQUATION_
                       'FSMAT#BUILDER_
    version,           'version number of current format
```

```
docType,
                 'version number of original document
original_version,
                 'last version capable of reading this file
minimum_version,
content_hint     'arbitrary hint string
```

The values for grp_access and all_access are as follows:

0	No read or write permissions for the file. (default)
1	Read permission for the file
2	Read and write permissions for the file.

See also READ_DATA_FILE@

WRITE_FILE@

Writes a string to an open operating system file

Format WRITE_FILE@(file, string)

Arguments

file	The full path name of the file opened by OPEN_ASCII_FILE@.
string	The string to be written to the specified operating system file.

Description Writes data to an open file. Before writing to file, you must open it using OPEN_ASCII_FILE@. After writing to file, you should close it using CLOSE_FILE@. If file was opened in "write" mode by OPEN_ASCII_FILE@, then string overwrites the contents of file. If file was opened in "append" mode by OPEN_ASCII_FILE@, then string is appended to the end of file.

While the write is in progress, WRITE_BINARY_FILE@ places a write lock on the file. This lock is cleared when the file write is complete. If two ELF macros try to write the same file simultanously, one of the ELF macros will throw an error indicating that the file is locked.

See also OPEN_ASCII_FILE@
CLOSE_FILE@

WRITE_GRAPHIC_BUFFER@

Writes the graphic onto a buffer

Format data = WRITE_GRAPHIC_BUFFER@(gfx)

Arguments	gfx	A graphics handle.

Description Writes the graphic data associated with gfx into a format suitable for writing to a file. This data is a long ASCII string.

WRITE_GRAPHIC_FILE@

Writes the graphic to a file

Format WRITE_GRAPHIC_FILE@(gfx, path name)

Arguments	gfx	A graphics handle.
	path name	The name of the file to which the graphic will be written. If you will want to read the file into the Graphics Editor, you must append an .ag suffix to the file name.

Description Writes the graphic data associated with the gfx graphics handle in standard Applix*ware* graphics format.

While the write is in progress, WRITE_GRAPHIC_FILE@ places a write lock on the file. This lock is cleared when the file write is complete. If two ELF macros try to write the same file simultanously, one of the ELF macros will throw an error indicating that the file is locked.

WWW_FETCH_URL@

Retrieves a file from a Web site

Format WWW_FETCH_URL@(url, filename)

Arguments	url	The URL (Uniform Resource Locator) of the file being retrieved.
	filename	The name of the file into which the retrieved information is written. The format of this file depends on the type of file retrieved. For example, if you are retrieving a GIF file, filename will be a GIF file. This macro does not convert information from one format to another.

Description Retrieves a file stored on the World Wide Web. If you are running a proxy server, the name of this server (and port number, if necessary) are stored in an Applix*ware* profile variable named axUrlProxyServer.

If you wish to use your own macro for retrieving data, store the name of this macro in an Applix*ware* profile variable named axUrlFetchMacro. The macro you create is called using the same url and filename arguments passed to this macro.

NOTE: Under NT, this macro uses the LineMode program supplied directly from CERN source files. You must have a LineMode executable in your path.

See also FILTER_HTML_TO_WP@
FILTER_URL_TO_GR@
FILTER_URL_TO_WP@
WP_IMPORT_HTML@
WP_IMPORT_URL@
WWW_MERGE_URL@

WWW_MERGE_URL@

Converts a partial URL into a full URL

Format URL = WWW_MERGE_URL@(partial, base)

Arguments

partial	A right hand portion of a URL; this is either a file name or a relative file name.
base	The base portion of the URL.

Description Merges the two portions of the URL into one component. The returned URL can then be used to retrieve information from the WEB.

In many cases, the two arguments are simply joined together after insuring that a foreslash ("/") separates them. This macro is most often used when you have obtained a base URL from a <BASE> tag or from a previous document. You would then use this macro to combine the components together.

WWW_MERGE_URL@ understands the different parts of the URL sufficiently so that it can tell where and how to merge the arguments.

See also FILTER_HTML_TO_WP@
FILTER_URL_TO_GR@
FILTER_URL_TO_WP@
WP_IMPORT_HTML@
WP_IMPORT_URL@
WWW_FETCH_URL@

WWW_PARSE_URL@

Sepparates a URL into parts

Format WWW_PARSE_URL@(urlString)

Arguments

urlString	a full URL containing a protocol, a server name, and a document.

Description Returns a www_url_parts@ format. This format contains the following fields:

```
format www_url_parts@
scheme,                    ' Trailing ':' is clipped
net_loc,                   ' Leading '//' is clipped
path,           ' Leading '/', if present, is NOT clipped
params,         ' Leading ';' is clipped
query,          ' Leading '?' is clipped
fragment        ' Leading '#' is clipped
```

For more information, see R. Fielding's document "Relative Uniform Resource Locators" at http://ds.internic.net/rfc/rfc1808.txt.

See also WWW_RECOMBINE_URL@

WWW_RECOMBINE_URL@

Builds a URL from its parts

Format WWW_RECOMBINE_URL@(urlParts)

Arguments

urlParts	a www_url_Parts@ format.

Description Returns a string containing a URL built from the supplied parts. The urlParts argument is a www_url_parts@ format. This format contains the following fields:

```
format www_url_parts@
scheme,                    ' Trailing ':' is clipped
net_loc,                   ' Leading '//' is clipped
path,           ' Leading '/', if present, is NOT clipped
params,         ' Leading ';' is clipped
query,          ' Leading '?' is clipped
fragment        ' Leading '#' is clipped
```

For more information, see R. Fielding's document "Relative Uniform Resource Locators" at http://ds.internic.net/rfc/rfc1808.txt.

See also WWW_MERGE_URL@

XKEYS_LIST@

Returns an array of the names and keysym numbers of keyboard keys

Format keyArray = XKEYS_LIST@()

Description Returns an array in which each array element is a two-element sub-array. The first element of the sub-array is the name of a keyboard key. The second element is the *keysym* number assigned to the keyboard key.

XKEYS_LIST@ returns information on all keys recognized by Applix*ware* as indicated in the table in the file errmsg4.help and any keys that have been defined using DEFINE_KEY@.

See also DEFINE_KEY@

YEAR@

Extracts the year from a serial date number

Format YEAR@(dateNumber)

Arguments dateNumber A serial date number.

Description The YEAR@ function extracts the year (00-199) from a serial date number. You can enter a serial date as an argument for the YEAR@ function. The formula YEAR@(30899) returns 84.

You can also use the DATE@ or TODAY@ function as an argument for the YEAR@ function. For example, the formula YEAR@(DATE@(84,1,1)) returns 84.

YEARABS@

Extracts the the absolute year from a serial date number

Format YEARABS@(dateNumber)

Arguments dateNumber A serial date number.

Description The YEARABS@ function extracts the absolute year (e.g., 1999) from a serial date number. You can enter a serial date as an argument for the YEARABS@ function. The formula YEARABS@(30899) returns 1984.

You can also use the **DATE@** or **TODAY@** function as an argument for the YEAR-ABS@ function. For example, the formula YEARABS@(DATE@(84,1,1)) returns 1984.

YEARFRAC@

Calculate the fraction of a year between two dates

Format YEARFRAC@(startDate, endDate, basis)

Arguments

startDate	The date value of the date at the beginning of the period to be measured.
endDate	The date value of the date at the end of the period to be measured.
basis	The type of day count basis to use.

Description YEARFRAC@ returns the fraction of a calendar year between the date values startDate and endDate. A date value is the number of calendar days after 12/31/1899 that the given date falls. This number is returned by the macro DATEVALUE@. For example, DATEVALUE@("1/1/1900") is 1, and DATEVALUE@("12/31/1995") is 35,430.

The basis argument is the type of day count basis to use:

Basis	Day count basis
0 or omitted	US (NASD) 30/360 basis
1	actual/actual
2	actual/360
3	actual/365
4	European 30/360

A few examples follow:

YEARFRAC@(DATEVALUE@("01/01/95"),DATEVALUE("06/30/95"), 0)

This calculate the fraction of a calendar year based on 30-day months and a 360-day year. This yields 0.5 years.

YEARFRAC@(DATEVALUE@("01/01/95"),DATEVALUE@("06/30/95"),2)

This calculates the fraction of a calendar year between January 1, 1995 and June 30, 1995, based on an actual day count, and a 360-day year. This yields 0.5027778 years.

See also DATEVALUE@

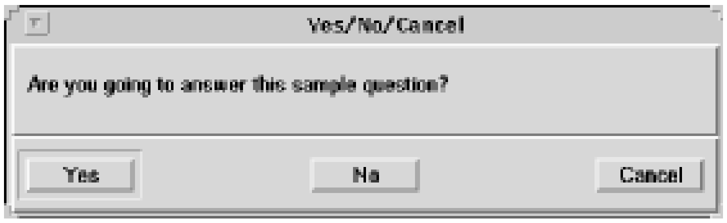
YES_NO_CANCEL_PROMPT@

Displays a dialog box with a label and YES, NO, and Cancel push buttons

Format flag =YES_NO_CANCEL_PROMPT@(label[, title])

Arguments	label	The text you want to display in the dialog box. The text should be able to be answered with either "yes" or "no."
	title	The name to be assigned this dialog box. This text is optional. If it is omitted, the string "Yes/No" is displayed.

Description Displays a dialog box displaying the text of label. This label should be asking a yes/no question.



Are you going to answer this sample question?

At this point, the user can press one of the three buttons to answer the question. This macro returns TRUE if YES is selected and FALSE if NO is selected. If Cancel is selected, the dialog box is exited and nothing is returned.

See also YES_NO_PROMPT@
YES_NO_HELP_PROMPT@

YES_NO_HELP_PROMPT@

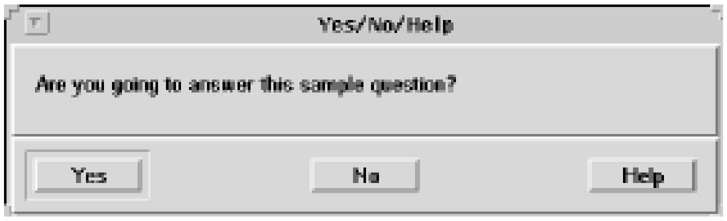
Displays a dialog box with YES, NO, and Help push buttons

Format flag = YES_NO_HELP_PROMPT@(label, title, helpTopic)

Arguments	label	The text you want to display in the dialog box. The text should be able to be answered with either "yes" or "no."

title	The text of the title that should appear in the dialog box.
helpTopic	The help topic id. This id can be a string.

Description Displays a dialog box displaying the text of label. This label should be asking a yes/no question.


Yes/No/Help

Are you going to answer this sample question?

At this point, the user can press one of the two buttons to answer the question. This macro returns TRUE if YES is selected; it returns FALSE if NO is selected.

See also YES_NO_CANCEL_PROMPT@

YES_NO_PROMPT@

YES_NO_PROMPT@

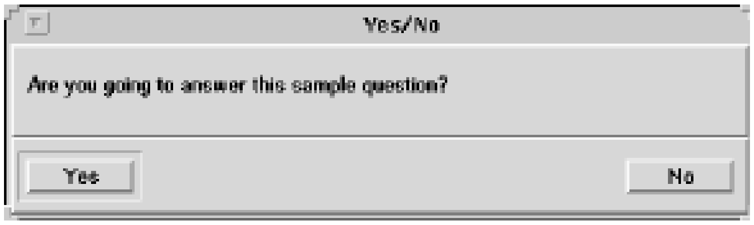
Displays a dialog box with YES and NO push buttons

Format flag = YES_NO_PROMPT@(label[, title])

Arguments

label	The text you want to display in the dialog box. The text should be able to be answered with either "yes" or "no."
title	The name to be assigned this dialog box. This text is optional. If it is omitted, the string "Yes/No" is displayed.

Description Displays a dialog box displaying the text of label. This label should be asking a yes/no question.


Yes/No

Are you going to answer this sample question?

At this point, the user can press one of the two buttons to answer the question. This macro returns TRUE if YES is selected; it returns FALSE if NO is selected.

See also YES_NO_CANCEL_PROMPT@

YES_NO_HELP_PROMPT@

YES_NO_HELP Built-in Dialog box

No help is available for this dialog box.