Applixware ELF
Example Macros

This manual was produced using Applixware.

Printed: June 2010

axnet Example	This help topic contains several axnet example macros.
ax_array Example	This macro is an example of maintaining an array.
ax_back Example	This macro copies or moves a specified file to another directory.
ax_char Example	This macro takes an ASCII character and returns the character number and the corresponding character in Symbol font.
ax_command Example	This macro posts an info message with the command line options used when Applix*ware* was last invoked.
ax_cursr Example	This macro posts a dialog box with a list of cursor symbols.
ax_doc Example	This macro displays a dialog box that lets you open or print a document, or exit Applix*ware*.
ax_env Example	This macro retrieves environment and display information.
ax_file Example	This macro displays files and directories in a dialog box.
ax_graph Example	This macro uses stroke macros to create Graphics objects in a Graphics window.
ax_help Example	This macro uses **HELP_HYPERLINK@** to bring up an ELF help topic, then copies the topic to a Words document.
ax_pref Example	This macro is an example of changing preferences.
ax_shell Example	This macro executes an operating system command and displays the result of the command in a Words document.
ax_spell Example	This macro creates an array of words from a comma separated string, then performs a spellcheck of the words.

ax_table Example	This macro posts a dialog box with the line numbers of profile settings in your ax_prof4 file.
ax_task Example	This macro is an example using the **NEW_TASK@** and **PEND_FOR_NEW_TASK@** macros.
ax_time Example	This macro posts a dialog box with the current date and time.
ax_window Example	This macro retrieves the window number and title of the current window.
Mail Rule Macro	This macro checks incoming messages for an "urgent" flag or the word "urgent" in the subject, placing the message in a folder named "Urgent."
SS_MAKE_ERROR_DATA@ Macro Example	This macro contains the macro SS_MAKE_ERROR_DATA@.
COMPOSE_TIME@ Macro Example	This macro contains the macros COMPOSE_TIME@ and DECOMPOSE_TIME@, and shows how to move between UNIX time format and the Applix*ware* Date_Time_Array_ format.
NEW_TASK@ Example	This macro shows how tasks started with New_Task@ can pass control of the execution thread using the DELAY@ macro.
PEND_FOR_NEW_TASK@ Example	This macro shows how to start a task with PEND_FOR_NEW_TASK@. Note that the new task runs to completion before control is returned to the Parent task.
DB_CTRL_RIGHT_MOUSE_MENU@ Example	This macro shows how to create and display menus for ELF dialog box controls, such as edit b oxes and entry boxes.

DB_CTRL_INSET@ Example	This macro shows how to display a file in a dialog box.
IS_ASCII_FILE@ Example	This macro presents a list of files and directories ELF dialog box controls, such as edit boxes and entry boxes.

Axnet Example

```
/*****************************************************************
 * AXNET_SERVER
 *       Create server "demo", respond to simple messages
 *       Note the real service name is "<username>:demo".
 *****************************************************************/
macro AXNET_SERVER
        var info

        info = "demo_macro","rpc_pong"
        AXNET_SERVICE_REGISTER@("demo",info)
endmacro

/*****************************************************************
 * AXNET_DROP_SERVER
 *       Convenience function for dropping the server
 *****************************************************************/
macro AXNET_DROP_SERVER
        AXNET_SERVICE_UNREGISTER@("demo")
endmacro


/*****************************************************************
 * DEMO_MACRO
 *       The Server side function. This function will
 *       respond to one of three "messages":
 *               ping, pong, or file.
 *       Anything else is an error.
 *****************************************************************/
macro DEMO_MACRO(cmd,data1,data2)
        if cmd = "ping"
                return
        if cmd = "pong"
                return(data1)
```

```
        if cmd = "file"
        {
                WRITE_BINARY_FILE@(data1,data2)
                return(data2)
        }
        ERROR@(1,"Not a known demo command",cmd)
endmacro

/*****************************************************************
 * AXNET_CLIENT
 *       Send messages to the demo server
 *****************************************************************/
macro AXNET_CLIENT(hostname,username)
        var service
        var stuff
        var result

        service = username ++ ":demo"
'
' Send message "ping". There is no response.
'
        AXNET_RPC@(hostname,service,"demo_macro","ping")

'
' Send message "pong". The result should be the name
' of the service.
'
        result = AXNET_RPC@(hostname, service,
                        "demo_macro","pong",service)
        if result <> service
                error@("pong failed")

'
' Send message "pong". The result should be the string
' sent as the fourth argument.
'
        result = AXNET_RPC@(hostname,service,
                        "rpc_pong","Hello World!")
        if result <> "Hello World!"
                error@("rpc_pong failed")

'
' Send message "file". The file should be created.
' The information written to the file should be
```

```
' returned.
'
        stuff = READ_BINARY_FILE@("/etc/passwd")
        result = AXNET_RPC@(hostname,service,
                        "demo_macro","file",
                        "/tmp/zztest",stuff)
        if result <> stuff
                error@("file xfer failed")

'
' Send message "make_error". An error message should be
' thrown.
'
        stuff = AXNET_RPC@(hostname, service,
                        "demo_macro","make_error")
endmacro

/*****************************************************************
 * RPC_PONG
 *      Simply returns the string sent to it
 *****************************************************************/
macro RPC_PONG(str)
        return(str)
endmacro
```

For more information see:

axnet Menu

ax_array Example

```
'  This macro is an example of maintaining an array.
'  Two arrays are initialized with values, then they
'  are modified through a dialog box.  The dialog box is
'  an interface for adding and deleting array items.
'  Select Tools Ý Compile & Run to run this macro.


macro ax_array
     var dbox, exit_ctrl
     var array2,array3, x, size
     var fruit_enter, type_enter, insert_point
     var index, temp_array, position

     dbox = DB_LOAD@("ax_array.d")
     DB_CTRL_RETURN_ON_CHANGE@(dbox,"Add",TRUE)
     DB_CTRL_RETURN_ON_CHANGE@(dbox,"Delete",TRUE)
     DB_CTRL_RETURN_ON_CHANGE@(dbox,"Exit",TRUE)
     DB_CTRL_RETURN_ON_CHANGE@(dbox,"fruit",TRUE)
     DB_CTRL_RETURN_ON_CHANGE@(dbox,"type",TRUE)

' Create and initialize arrays
     array3 = "Apple, Cherry, Grape, Orange, Pear"
     array3 = ARRAY_FROM_STRING@(array3, ",")
     array2 = "Macintosh, Bing, Red, Mandarin, Bosc"
     array2 = ARRAY_FROM_STRING@(array2, ",")

 while (not DB_CANCELLED@(dbox))
     DB_CTRL_STRINGS@(dbox, "fruit", array3)
     DB_CTRL_STRINGS@(dbox, "type", array2)
     size = ARRAY_SIZE@(array3)                    ' get size of array
     DB_CTRL_VALUE@(dbox, "number", size)
     DB_DISPLAY@(dbox)
     exit_ctrl = DB_EXIT_CTRL@(dbox)

'    Check the exit_ctrl condition
     case of (exit_ctrl)
     case "Add"
               fruit_enter = DB_CTRL_GET_VALUE@(dbox, "enter_f")
               type_enter = DB_CTRL_GET_VALUE@(dbox, "enter_t")
               insert_point = DB_CTRL_GET_VALUE@(dbox,
```

```
                                    "InsertPoint")
        'insert fruit and type in arrays
        if ( fruit_enter <> "" )      ' Check if fruit_enter is not NULL
        {
                if ( insert_point = 0)       ' Insert in Order
                {
                    array3 = ARRAY_INSERT_ORDERED@(array3,
                                fruit_enter, TRUE, TRUE)
                    index = ARRAY_INDEX@(array3, fruit_enter)
                    array2 = ARRAY_INSERT@(array2, type_enter,
                                index)
                }
                else if ( insert_point = 1)
                'Insert at beginning of array
                {
                    size = ARRAY_SIZE@(array3)
                    temp_array[0] = fruit_enter
                    array3 = ARRAY_APPEND@(temp_array, array3)
                    temp_array[0] = type_enter
                    array2 = ARRAY_APPEND@(temp_array,array2)
                }
                else                 'Insert at end of array
                {
                    size = ARRAY_SIZE@(array3)
                    array3 = ARRAY_INSERT@(array3,
                                fruit_enter,size)
                    array2 = ARRAY_INSERT@(array2,type_enter,
                                 size)
                }
        }

case "Delete"     ' Delete an array element
        IF (size <>0 )
        {
                index = DB_CTRL_GET_VALUE@(dbox, "fruit")
                array3 = ARRAY_DELETE@(array3, index)
                array2 = ARRAY_DELETE@(array2, index)
        }

case "fruit"         ' Set selection in both list boxes
        index = DB_CTRL_GET_VALUE@(dbox, "fruit")
        DB_CTRL_VALUE@(dbox, "type", index)

case "type"
```

```
            index = DB_CTRL_GET_VALUE@(dbox, "type")
            DB_CTRL_VALUE@(dbox, "fruit", index)
    endcase
  wend
 endmacro
```

ax_back Example

```
'  This macro copies or moves a specified file to another directory
'  You have the option of copying or moving the file to the current
'  directory, or specifying a different directory.
'  The macro creates the directory if it does not exist.
'  Select Tools Ý Compile & Run to run this macro.

macro ax_back
    var dir, file, current
    var dbox, exit_ctrl, choice

    current = CURRENT_DIR@()            ' get current directory
    dbox = DB_LOAD@("ax_back.d")
    DB_CTRL_RETURN_ON_CHANGE@(dbox,"OK",TRUE)
    DB_CTRL_VALUE@(dbox,"directory",current)
    DB_DISPLAY@(dbox)
    exit_ctrl = DB_EXIT_CTRL@(dbox)
    IF DB_CANCELLED@(dbox)
        RETURN
    IF exit_ctrl = "OK"
    {
        file = DB_CTRL_GET_VALUE@(dbox,"file_name")
        IF ( FILE_EXISTS@(file) = FALSE)
        {
            INFO_MESSAGE@("File " ++ file ++" does not exist")
            RETURN
        }
        dir = DB_CTRL_GET_VALUE@(dbox,"directory")
        IF (DIR_EXISTS@(dir) = FALSE)
            CREATE_DIR@(dir)
        choice = DB_CTRL_GET_VALUE@(dbox,"choice")
        IF choice ' Move file
        {
            MOVE_FILE@(file, dir)
            INFO_MESSAGE@("Moved file " ++ file ++
```

```
                    " to directory " ++ dir )
        }
        ELSE      ' Copy file
        {
            COPY_FILE@(file, dir)
            INFO_MESSAGE@("Copied file " ++ file ++ " to directory "
                        ++ dir )
        }
    }
endmacro
```

ax_char Example

```
'  This macro takes an ASCII character and returns
'  the character number and the corresponding character
'  in Symbol font.
'  Select Tools Ý Compile & Run to run this macro.


macro ax_char
    var     dbox, exit_ctrl, num, font

    font = DB_XLATE_FONT@("Symbol", TRUE,TRUE)

    ' Post dbox
    dbox = DB_LOAD@("ax_char.d")
    DB_CTRL_RETURN_ON_CHANGE@(dbox, "Apply", TRUE)

    while (not DB_CANCELLED@(dbox))
            DB_DISPLAY@(dbox)
            exit_ctrl = DB_EXIT_CTRL@(dbox)
            IF exit_ctrl = "Apply"
            'Get character number, post symbol
            {
                    num = STRING_TO_NUM@(
                        DB_CTRL_GET_VALUE@(dbox, "character"))
                    DB_CTRL_VALUE@(dbox, "number", num)
                    DB_CTRL_TEXT_AND_FONT@(dbox,"symbol",
                                NUM_TO_STRING@(num), font)
                    DB_CTRL_DISPLAY@(dbox, "symbol", TRUE)
            }
    wend
endmacro
```

ax_command Example

```
'  This macro posts an info message with the command line options
'   used when Applixware was last invoked.
'  Select Tools Ý Compile & Run to run this macro.

macro ax_commd
    var     array, string

    array = COMMAND_LINE_OPTIONS@()
    BEEP@()
    if ( ARRAY_SIZE@(array) <> 0 )
    {
            string = ARRAY_TO_STRING@(array, ", ")
            INFO_MESSAGE@("The command line options used :\n"
                            ++ string)
    }
    else            'No options
            INFO_MESSAGE@("No command line options used.")
endmacro
```

ax_cursr Example

```
'  This macro posts a dialog box with a list of cursor symbols.
'  Choose a symbol from the list and click "OK"  to change the
'  pointer cursor.
'  Select Tools Ý Compile & Run to run this macro.


macro ax_cursr
    var     dbox, listcur, pointer, exit_ctrl

    dbox = DB_LOAD@("ax_cursr.d")
    GET_CURSOR_LIST@(listcur)
    DB_CTRL_STRINGS@(dbox, "pointer", listcur)
    DB_DISPLAY@(dbox)
    if DB_CANCELLED@(dbox)
```

```
            RETURN
    pointer = DB_CTRL_GET_VALUE@(dbox, "pointer")
    SET_LOOK_FEEL_VALUES@(70,500,1,0,TRUE,listcur[pointer],
                                    "watch",TRUE,0,"64x64")
endmacro
```

ax_doc Example

```
'  This macro displays a dialog box that lets you
'  open or print a document, or exit Applixware
'  Select Tools Ý Compile & Run to run this macro.


macro ax_doc
    var     dbox, exit_ctrl, choice,  print, open, background
    varfile, class, pclass, printers

    dbox = DB_LOAD@("ax_docs.d")
    DB_CTRL_RETURN_ON_CHANGE@(dbox,"OK",TRUE)
    DB_CTRL_RETURN_ON_CHANGE@(dbox,"doc_op",TRUE)
    DB_WINDOW_REMAIN@(dbox, TRUE)
    DB_CTRL_STRINGS@(dbox,"print_list",
                            LIST_OF_PRINTERS@())
    pclass[0] = "PostScript"
    pclass[1] = "PCL5"

    WHILE  NOT DB_CANCELLED@(dbox)
        DB_DISPLAY@(dbox)
        exit_ctrl = DB_EXIT_CTRL@(dbox)
        case of (exit_ctrl)
        case "doc_op"
        '  Set the gray state of controls for given choices
            choice = DB_CTRL_GET_VALUE@(dbox, "doc_op")
            case of (choice)
            case 0                      'Open
                DB_CTRL_GRAYED@(dbox,"document", FALSE)
                DB_CTRL_GRAYED@(dbox,"background", TRUE)
                DB_CTRL_GRAYED@(dbox,"exit", TRUE)
                DB_CTRL_GRAYED@(dbox,"print_list", TRUE)
                DB_CTRL_GRAYED@(dbox,"print_class", TRUE)

            case 1                      'Print
                DB_CTRL_GRAYED@(dbox,"document", FALSE)
```

```
        DB_CTRL_GRAYED@(dbox,"background",
                        FALSE)
        DB_CTRL_GRAYED@(dbox,"exit", TRUE)
        DB_CTRL_GRAYED@(dbox,"print_list", FALSE)
        DB_CTRL_GRAYED@(dbox,"print_class", FALSE)

    case 2                          'Exit Applixware
        DB_CTRL_GRAYED@(dbox,"document", FALSE)
        DB_CTRL_GRAYED@(dbox,"background", TRUE)
        DB_CTRL_GRAYED@(dbox,"exit", FALSE)
        DB_CTRL_GRAYED@(dbox,"print_list", TRUE)
        DB_CTRL_GRAYED@(dbox,"print_class", TRUE)
    endcase
case "OK"
    file = DB_CTRL_GET_VALUE@(dbox,"document")
    choice = DB_CTRL_GET_VALUE@(dbox, "doc_op")
    case of (choice)
    case 0      ' Open a document
        if FILE_EXISTS@(file)
            OPEN_DOC@(file)
        else
            INFO_MESSAGE@("Document " ++ file ++
                          " does not exist")
        return
    case 1      ' Print a document
        IF FILE_EXISTS@(file)
        {
            printers = DB_CTRL_GET_STRINGS@
                       (dbox, "print_list")
            print = DB_CTRL_GET_VALUE@
                       (dbox,  "print_list")
            class = pclass[DB_CTRL_GET_VALUE@
                                    (dbox,"print_class")]
            if  DB_CTRL_GET_VALUE@
                       (dbox, "background")
                PRINT_FILE_BACKGROUND@(file,printers[print],
                       FALSE,1,FALSE,TRUE,1,1,NULL, NULL,
                       NULL, AXHOME_DIR@() ++ "doc.ps",
                       NULL,NULL,NULL,NULL,NULL,class,
                       FALSE)
            else          'Do not print in background
                PRINT_FILE@(file,printers[print],FALSE,
                       1,FALSE,TRUE,1,1,NULL,NULL,NULL,
                       AXHOME_DIR@() ++  "doc.ps",NULL,
```

```
                                NULL,NULL,NULL,NULL,class, FALSE)
                }
                else
                    INFO_MESSAGE@("Document " ++ file ++
                                    " does not exist")
            case 2      'Exit Applixware
                if  DB_CTRL_GET_VALUE@(dbox, "exit")
                    EXIT_ALL@()     ' Exit with Save option
                else
                    LOGOUT@()               ' Exit without saving
            endcase
        endcase
    WEND
endmacro
```

ax_env Example

```
'  This macro retrieves environment and display information,
'  then posts the information in a dialog box.
'  Select Tools Ý Compile & Run to run this macro.

macro ax_env
    var     display, home, height, width, pid, machine, mach_choice
    var     dbox, audio, sys5, sys_dir, lang_dir, temp_dir, string
    varunit, number_style, x_pos
    string = "NULL,NULL,SUN4 or Solaris,HP300,NULL,MIPS,"++
                "SCO   UNIX,SGI,HP800,RIOS,NULL,NULL,M_88K,"++
                "CLIX Intergraph"
    ' Get information
    display = GET_ENV_VAR@("DISPLAY")
    home = USER_DIR@()
    ' USER_DIR@ same as GET_ENV_VAR@("HOME")

    height = GET_DISPLAY_HEIGHT@()
    width = GET_DISPLAY_WIDTH@()
    audio = SUPPORTS_AUDIO@()
    IF audio = 0
        audio = "No"
    ELSE
        audio = "Yes"
    sys5 = SYS5@()
    IF sys5 = 0
```

```
    sys5 = "No"
ELSE
    sys5 = "Yes"
sys_dir = SYSTEM_DIR@()
lang_dir = SYSTEM_LANG_DIR@()
temp_dir = TEMP_DIR@()
pid = UNIX_PROCESS_ID@()

machine = MACHINE_TYPE@()
mach_choice = ARRAY_FROM_STRING@( string,",")
mach_choice = ARRAY_INSERT@( mach_choice,"SVR4/USL",98)
mach_choice = ARRAY_INSERT@
        ( mach_choice,"Solaris x86", 99)

IF METRIC@()            ' Similar to METRICSYSTEM@
    unit = "Metric"
ELSE
    unit = "Inches"
IF NUMBERSTYLE@()
    number_style = "European"
ELSE
    number_style = "American"

' Post dialog box with info
x_pos = 190
dbox = DB_LOAD@("ax_env.d")
' Create labels with information
DB_CREATE_CTRL@(dbox, 5, "home", home, x_pos, 28)
DB_CREATE_CTRL@(dbox, 5, "display", display, x_pos, 50)
DB_CREATE_CTRL@(dbox, 5, "height", height, x_pos, 72)
DB_CREATE_CTRL@(dbox, 5, "width", width, x_pos, 94)
DB_CREATE_CTRL@(dbox, 5, "audio", audio, x_pos, 116)
DB_CREATE_CTRL@(dbox, 5, "sys5", sys5, x_pos, 138)
DB_CREATE_CTRL@(dbox, 5, "sys_dir", sys_dir, x_pos, 160)
DB_CREATE_CTRL@(dbox, 5, "lang_dir", lang_dir, x_pos, 182)
DB_CREATE_CTRL@(dbox, 5, "temp_dir", temp_dir, x_pos, 204)
DB_CREATE_CTRL@(dbox, 5, "pid", pid, x_pos, 226)
DB_CREATE_CTRL@(dbox, 5, machine ,
            mach_choice[machine], x_pos, 248)
DB_CREATE_CTRL@(dbox, 5, "unit" ,unit, x_pos, 270)
DB_CREATE_CTRL@(dbox, 5, "num_style" ,number_style,
                        x_pos,292)
DB_DISPLAY@(dbox)
IF DB_CANCELLED@(dbox)
```

```
        RETURN
endmacro
```

ax_file Example

```
'  This macro displays files and directories in a dialog box.
'  You can choose a file to learn its file size and permissions.
'  You can change directories with the Directory option.  You
'  can double-click on a directory in the Directories list area,
'  or you can choose a directory and click on Apply.
'  Select Tools Ý Compile & Run to run this macro.

macro ax_file
    var     dbox, exit_ctrl
    var     current, files, dirs
    var     choice, size, array, path
    var     index, parse,path_part, i, dir_part
    var     toggles, x, click_flag

    dbox = DB_LOAD@("ax_file.d")

    '  enable double-clicks on directories list
    DB_CTRL_PICK_DEFAULT@(dbox, "dir_list", TRUE)
    DB_CTRL_RETURN_ON_CHANGE@(dbox, "directory", TRUE)
    DB_CTRL_RETURN_ON_CHANGE@(dbox, "dir_list", TRUE)
    DB_CTRL_RETURN_ON_CHANGE@(dbox, "file_list", TRUE)

reset:
    ON ERROR
    {
            ERROR_BOX@()
            goto reset          ' Reset error handler
    }

    '  Initialize values for dbox
    files = SORT@(LIST_OF_FILES@(CURRENT_DIR@()))
    dirs = SORT@(LIST_OF_DIRS@(CURRENT_DIR@()))
    parse = CURRENT_DIR@()
    i = 0

    '  Break down pathname of current dir for option button
    WHILE parse <> NULL and parse <> "/"
```

```
        PARSE_PATHNAME@(parse,parse,path_part)
        path[i] = path_part
        i = i+1
WEND

DB_CTRL_STRINGS@(dbox, "directory", path)
DB_CTRL_STRINGS@(dbox, "file_list", files)
DB_CTRL_STRINGS@(dbox, "dir_list", dirs)

while not DB_CANCELLED@(dbox)
        DB_DISPLAY@(dbox)
        exit_ctrl = DB_EXIT_CTRL@(dbox)
        case of (exit_ctrl)
        case "file_list"     'File chosen, get size
                choice = DB_CTRL_GET_VALUE@(dbox, "file_list")
                files = DB_CTRL_GET_STRINGS@(dbox,"file_list")
                '  Get size of chosen file
                size = FILE_SIZE@(CURRENT_DIR@() ++ "/"
                                ++ files[choice])

                '  Initialize toggle button values
                DB_CTRL_VALUE@(dbox, "file_size", size)
                DB_CTRL_VALUE@(dbox, "read", FALSE)
                DB_CTRL_VALUE@(dbox, "write", FALSE)
                DB_CTRL_VALUE@(dbox, "execute", FALSE)

                '  get file info for user permissions
                array = FILE_SYSTEM_INFO@(CURRENT_DIR@() ,
                                files[choice], FALSE,FALSE,FALSE,
                                 TRUE, FALSE)
                for index = 0 to ARRAY_SIZE@(array)-1
                    if (array[index,1]<> -1)
                    {
                      if SUBSTRING@(array[index,5],2,1) = "r"
                            DB_CTRL_VALUE@(dbox, "read", TRUE)
                      if SUBSTRING@(array[index,5],3,1) = "w"
                            DB_CTRL_VALUE@(dbox, "write", TRUE)
                      if SUBSTRING@(array[index,5],4,1) = "x"
                            DB_CTRL_VALUE@(dbox, "execute",
                                                TRUE)
                    }
                next index

        case   "directory"          'Possible change in directory option
```

```
        '  Get all values in option button
        toggles = DB_CTRL_GET_STRINGS@(dbox,
                                        "directory")

        '  Get option button choice
        choice = DB_CTRL_GET_VALUE@(dbox,"directory")
        if choice <> 0              '  Change in directory
        {
            toggles = SUBARRAY@(toggles, choice)
            x = (ARRAY_SIZE@(toggles)-1)

            '  Put together path of directory to set current_dir
            path = NULL
            for index = 0 to  (ARRAY_SIZE@(toggles)-1)
                    path = path++"/"++toggles[x]
                    x = x-1
            next index

            SET_CURRENT_DIR@(path)
            DB_CTRL_STRINGS@(dbox,"file_list",
                    SORT@(LIST_OF_FILES@(path)))
            DB_CTRL_STRINGS@(dbox,"dir_list",
                    SORT@(LIST_OF_DIRS@(path)))
            DB_CTRL_STRINGS@(dbox, "directory", toggles)
            DB_CTRL_VALUE@(dbox, "directory", 0)
        }

case   "dir_list"   ' Choice in Directories list area
        if( DB_CTRL_GET_VALUE@(dbox, "dir_list") <> -1)
            click_flag = true
        else
            click_flag = false

case "Apply"                '  Change to choice in Directories
        choice = DB_CTRL_GET_VALUE@(dbox, "dir_list")
        dirs = DB_CTRL_GET_STRINGS@(dbox,"dir_list")
        toggles = DB_CTRL_GET_STRINGS@(dbox,
                                        "directory")
        x = (ARRAY_SIZE@(toggles)-1)

        '  Put together path of directory to set current_dir
        path = NULL
        FOR index = 0 to  (ARRAY_SIZE@(toggles)-1)
            path = path++"/"++toggles[x]
```

```
                    x = x-1
                next index
                path = path++"/"++dirs[choice]

                toggles = ARRAY_INSERT@(toggles, dirs[choice], 0)
                SET_CURRENT_DIR@(path)
                DB_CTRL_STRINGS@(dbox,"file_list",
                        SORT@(LIST_OF_FILES@(path)))
                DB_CTRL_STRINGS@(dbox,"dir_list",
                        SORT@(LIST_OF_DIRS@(path)))
                DB_CTRL_STRINGS@(dbox, "directory", toggles)
                DB_CTRL_VALUE@(dbox, "directory", 0)
        endcase
    wend
endmacro
```

ax_graph Example

```
'  This macro uses stroke macros to create Graphics objects
'  in a Graphics window.
'  Select Tools Ý Compile & Run to run this macro.

macro ax_graph
    ' Open a graphics window
    OPEN_DOC@(USER_DIR@() ++ "/ax_graph.ag")
    DELAY@(2)

    ' Draw oval
    GR_TOOL_PICK@(5)
    STROKE_START@(10,10)
    STROKE_END@(100,100)

    ' Draw a free form object
    GR_TOOL_PICK@(6)
    STROKE_START@(140,31)
    STROKE_POINT@(145,129)
    STROKE_POINT@(150,120)
    STROKE_POINT@(155,31)
    STROKE_POINT@(157,38)
    STROKE_POINT@(163,135)
    STROKE_POINT@(167,139)
    STROKE_POINT@(174,31)
    STROKE_END@(200,31)
```

```
endmacro
```

ax_help Example

```
'  This macro uses HELP_HYPERLINK@ to bring up an
'  ELF help topic, then copies the topic to a
'  Words document.
'  Select Tools Ý Compile & Run to run this macro.


macro ax_help
    var     dbox, exit_ctrl, args, windows
    var     index, num

    '  call help on HELP_HYPERLINK@ macro
    args[0] = "HELP_HYPERLINK@"
    args[1] = "ELF"
    HELP_HYPERLINK@(args)

    '  Open a Words window
    OPEN_DOC@(TEMP_DIR@() ++ "/ax_help.aw")
    num = CURRENT_WINDOW_NUM@()

    ' See if Help window is still around
    windows = LIST_OF_WINDOWS@()
    for index = 0  to ARRAY_SIZE@(windows)
            if TRIM@(windows[index,1]) = "Applix Help"
            {
                    SELECT_WINDOW@(windows[index,0], TRUE)
                    goto continue
            }
    next index

    '  After finding Help window, copy topic to clipboard
continue:
    HELP_COPY_TOPIC@()

    '  Make Words window the selected window
    SELECT_WINDOW@(num, TRUE)
```

```
    '  Paste copied Help topic in Words window
    WP_PASTE@()
    REPAINT_ALL@()       ' Refresh all Applixware window displays
endmacro
```

ax_pref Example

```
'  This macro is an example of changing preferences.
'  The macro checks the type of printer setting in the
'  axPrintDefaultType preference and the name of the
'  default printer in the axPrintPrinter preference.
'  The preference settings are displayed in a dialog box,
'  and you can change the settings in the dialog box
'  to the desired values.  The macro changes and saves
'  the preferences.  The new settings take effect without
'  having to exit the product.
'  Select Tools Ý Compile & Run to run this macro.


macro ax_pref
    var  print_setting, target_printer, strings
    var axhome_dir, profile, dbox, exit_ctrl

    '  Initialize dialog box
    dbox = DB_LOAD@("ax_pref.d")
    DB_CTRL_RETURN_ON_CHANGE@(dbox,"OK",TRUE)

    'Get home directory
    axhome_dir = AXHOME_DIR@()

    ' make full pathname of ax_prof4 file
    profile = axhome_dir ++ "/ax_prof4"

    ' Check type of printer and default target printer
    print_setting = TRIM@(PREFERENCES@("axPrintDefaultType"))
    target_printer = TRIM@(PREFERENCES@("axPrintPrinter"))

    ' Set values for dbox display
    if print_setting = "PCL5"
            DB_CTRL_VALUE@(dbox, "type", 1)
    else            ' set to PostScript
            DB_CTRL_VALUE@(dbox, "type", 0)
```

```
    DB_CTRL_VALUE@(dbox,"printer",target_printer)
    DB_DISPLAY@(dbox)
    if DB_EXIT_CTRL@(dbox) = "OK"
    {       ' Get type of printer
            strings = DB_CTRL_GET_STRINGS@(dbox, "type")
            print_setting = strings[DB_CTRL_GET_VALUE@
                                    (dbox, "type")]

            ' Get name of default target printer
            target_printer = DB_CTRL_GET_VALUE@(dbox,"printer")

            '  Change and save the preferences
            CHANGE_PREFS@("axPrintDefaultType", print_setting)
            CHANGE_PREFS@("axPrintPrinter", target_printer)
            SAVE_PREFERENCES@(profile)
    }
endmacro
```

ax_shell Example

```
'  This macro executes an operating system command and displays the
'  result of the command in a Words document.  The macro
' TABS_TO_SPACES@ converts tabs in the output to spaces.
'  Select Tools Ý Compile & Run to run this macro.


macro ax_shell
    var     lines, size, i

    lines = SHELL_COMMAND@
                        (PROMPT@("Shell command to run"))
    WP_APPLICATION_DLG@(NULL)
    FOR i = 0 to ARRAY_SIZE@(lines)-1
            TYPE@(TABS_TO_SPACES@(lines[i]))
            WP_RETURN_KEY@()
    NEXT i
endmacro
```

ax_spell Example

```
'  This macro creates an array of words from a comma separated string,
'  then performs a spellcheck of the words.  The macro displays the
'  array of misspelled words with DUMP_ARRAY@.
'  Select Tools Ý Compile & Run to run this macro.


macro ax_spell
     var      array, misspelled_words, string

     '  Create an array
     array = COMMA_SPLIT@("bananaa, dog, incorrec, manyh, no")

     '  Get misspelled words
     misspelled_words = CHECK_SPELLING@(array)
     DUMP_ARRAY@(misspelled_words)
     string = COMMA_SEPARATE@(misspelled_words)
     INFO_MESSAGE@("The string of misspelled words is \n"
                          ++ string)
endmacro
```

ax_table Example

```
'  This macro posts a dialog box with the line numbers of profile
'   settings in your ax_prof4 file
'  Select Tools Ý Compile & Run to run this macro.


macro ax_table
     var      dbox, exit_ctrl, string
     var      array, headings, index

     '  call ax_creat macro
     array = ax_creat()
     dbox = DB_LOAD@("ax_2d.d")
     DB_TITLE@(dbox,"ax_prof4 Line Numbers")
     DB_CTRL_RETURN_ON_CHANGE@(dbox,"find", TRUE)
     DB_CTRL_DEFAULT_BUTTON@(dbox, "find", TRUE)
     '  Set table headings
     headings[0,0] = "Line #"
```

```
    headings[0,1] = 50
    headings[1,0] = "Profile"
    headings[1,1] = 175
    headings[2,0] = "Setting"
    headings[2,1] = 230

    '  Prepare dbox to receive table data
    DB_PAINT@(dbox)
    DB_TABLE_SET_DATA@(dbox, "Table", array, headings)
    while not DB_CANCELLED@(dbox)
            DB_DISPLAY@(dbox)
            exit_ctrl = DB_EXIT_CTRL@(dbox)
            if exit_ctrl = "find"          ' Find a given profile
            {
                    string = TRIM@(DB_CTRL_GET_VALUE@(dbox,
                                    "profile_name"))
                    index = LIST_FIND_ALPHA@
                                (ARRAY_COLUMN@(array,1), string)
                    if ( index = -1)
                        INFO_MESSAGE@(
                                "Not a valid profile name, try again.")
                    else        ' Scroll table so given profile is top row
                        DB_TABLE_SET_NEW_TOP_ROW@(dbox,
                                                        "Table", index)
            }
    wend
endmacro

'  This macro creates a 3 column array of profile settings and line
'  numbers for use in the ax_table macro

macro ax_creat
    var     profile_array, number_array, array_2D
    var     size, count, string

    '  Get contents of ax_prof4 as an array
    profile_array = ax_read(AXHOME_DIR@() ++ "/ax_prof4")
    size = ARRAY_SIZE@(profile_array)

    ' Create a 3 column array of profile settings and line number
    for count = 0 to size-1
            string = profile_array[count]
            array_2D[count, 0] = count+1
            array_2D[count, 1] =
```

```
                SUBSTRING@(string,1,
                                STRING_INDEX@(string,":")-1)
        array_2D[count, 2] = SUBSTRING@(string,
                                STRING_INDEX@(string,":")+1)
    next count
    return(array_2D)
endmacro

'  This macro opens an operating system file in read mode,
'  reads each line into an array, then closes the file

include "errors_.am"
macro ax_read(name)
    var     lines, count, file
    count = 0
    if (IS_NULL@(name) )
        file = PROMPT@("Filename")
    else
        file = name
    OPEN_ASCII_FILE@(file, "r")

    '  Error handling code, READ_FILE@ throws error
    '  when it reaches end of file
    ON ERROR
    {
        IF ERROR_NUMBER@() = ERR#CANNOT_READ_
        {
            CLOSE_FILE@(file)
            RETURN(lines)
        }
        ELSE
            ERROR_RETHROW@()
    }
    while IS_ASCII_FILE@(file)
        lines[count] = READ_FILE@(file)
        count = count + 1
    wend
    CLOSE_FILE@(file)
endmacro
```

ax_task Example

```
'    This macro is an example using the NEW_TASK@ and
```

```
'    PEND_FOR_NEW_TASK@macros.  If you choose the
'    NEW_TASK@ option, this macro continues to run, and the dbox
'    exits after a2-second delay.  If you choose the
'    PEND_FOR_NEW_TASK@ option, this macro waits for the
'    ax_task_child macro to complete, then continues operation.  The
'    dbox exits after a 2-second delay.
'   Select Tools Ý Compile & Run to run this macro.


macro ax_task
    var     dbox, exit_ctrl, choice, arguments, help

    dbox = DB_LOAD@("ax_task.d")
    DB_DISPLAY@(dbox)
    exit_ctrl = DB_EXIT_CTRL@(dbox)
    case of (exit_ctrl)
    case "Apply"
            choice = DB_CTRL_GET_VALUE@(dbox, "choice")
            if choice   ' Pend for new task
            {
                    arguments[0] = "a Pend For New Task"
                    PEND_FOR_NEW_TASK@("ax_task_child",
                                    arguments)
                    DELAY@(2)
            }
            else        ' New task
            {
                    arguments[0] = "a New Task"
                    NEW_TASK@("ax_task_child",arguments)
                    DELAY@(2)
            }
    endcase
    '  Check if ax_task_child is still displaying dbox
    IF PROMOTE_DIALOG@("ax_task Child Task")
            INFO_MESSAGE@("Parent Task Finished")
    RETURN
endmacro

'  This macro is part of the ax_task example
macro ax_task_child(arguments)
    var     dbox, promoted

    dbox = DB_LOAD@("ax_task1.d")
    DB_CREATE_CTRL@(dbox, 5, "Source", arguments[0], 44, 63)
```

```
    DB_DISPLAY@(dbox)
endmacro
```

ax_time Example

```
'  This macro posts a dialog box with the current date
'  and time.  The macro acts as a clock, displaying and
'  continually updating the hours, minutes, and seconds
'  in 24-hour format.
'  Select Tools Ý Compile & Run to run this macro.


include "datetim_.am"
macro ax_time
    var      dbox, date, get_time, time

    var format date_time_array_ timer
    get_time = CURRENT_TIME@()       ' Get current system time
    date = DATE_FORMAT@(get_time, 3001)  ' Format date
    dbox = DB_LOAD@("ax_time.d")
    DB_TIMER@(dbox, 1)
    DB_CREATE_CTRL@(dbox, 5, "time", time, 70, 15 )

    'Create time label
    DB_CREATE_CTRL@(dbox, 5, "date", date, 10, 40 )

    'Create date label
  WHILE NOT DB_CANCELLED@(dbox)
    timer = DECOMPOSE_TIME@(CURRENT_TIME@())
    if timer.hour < 10
            timer.hour = "0" ++ timer.hour
    if timer.minute < 10
            timer.minute = "0" ++ timer.minute
    if timer.second < 10
            timer.second = "0" ++ timer.second
    time = FORMAT@("%s:%s:%s",timer.hour,
                    timer.minute,timer.second)
    DB_CTRL_TITLE@(dbox, "time", time)
    DB_DISPLAY@(dbox)
  wend
```

endmacro

ax_window Example

```
'  This macro retrieves the window number and title of the current
'  window. If the current window is a Words document, the document
'  is saved under a new name, with the string "COPY" appended to the
'  title.
'  Select Tools Ý Compile & Run to run this macro.


include "app_ids_.am"
include "windows_.am"
include "wp_.am"
macro ax_window
    var     colNum, title, list, length, new_title
    var     format window_format@ window

    ' get current window number
    colNum = CURRENT_WINDOW_NUM@()
    if ( colNum <> -1 )                 ' Check if a window is selected
    {
            title = CURRENT_WINDOW_TITLE@()
            ' get window title
            window = WINDOW_INFO@(colNum)
            ' get window info
            if ( window.icon_id = APP#WORDS_ )
            ' Check if Words doc
            {
                    ' Get length of title
                    length = LEN@(title)
                    ' Chop off .aw
                    new_title = SUBSTRING@(title, 1, length-3)
                    new_title = new_title ++ "COPY.aw"
                    ' make sure we have window
                    SELECT_WINDOW@(colNum)
                    WP_SAVE_RENAME@(new_title)
            }
            else
                    INFO_MESSAGE@("Not a Words document.")
    }
    else
            INFO_MESSAGE@("No current Applixware window.")
```

```
endmacro
```

SS_MAKE_ERROR_DATA@ Example

```
'  This macro creates an error datum, then
'  tests the datum to see if it is an error datum.
'  Select Tools Ý Compile & Run to run this macro.


macro SS_MAKE_ERROR_DATA_x

    var bar

        SS_APPLICATION_DLG@()

        bar = SS_MAKE_ERROR_DATA@()

        If  SS_IS_ERROR@(bar) then
            info_message@("Bar is an Error Datum\n"
            ++"Press OK to Exit!")

    SS_EXIT@()


endmacro
```

COMPOSE_TIME@ Example

```
'  This macro shows how to convert data
'  between UNIX format and the
'  Applixware date_time_array_ format.
'  Select Tools Ý Compile & Run to run this macro.


include "datetim_.am"
macro Compose_Time_x

    var         time, get_time
    var         format date_time_array_ timer
```

```
    get_time = CURRENT_TIME@()       ' Get current system time

'       DECOMPOSE_TIME creates an array called timer. This array
'       can be passed to COMPOSE_TIME@ to return the number of
'       seconds since January 1, 1970, GMT.

    timer = DECOMPOSE_TIME@(CURRENT_TIME@())

'       COMPOSE_TIME takes the array timer, and returns the
'       number of seconds since January 1, 1970, GMT. This
'       example uses DATE_FORMAT@ to turn that number
'       into a formatted time.

    time = COMPOSE_TIME@(timer)

info_message@("The time is now "++DATE_FORMAT@(time,100))

endmacro
```

NEW_TASK@ Example

```
'  This macro shows how tasks started with New_Task@
'  can pass control of the execution thread using the DELAY@
'  macro.

macro NewTasks

var mac, x
mac = "children"
NEW_TASK@(mac)

for x = 1 to 100
    WORK_IN_PROGRESS@(x++"                          ", "Task1 has control...")

/*
*       The DELAY statement below passes control of the execution
*       thread to the children macro.
*/

    delay@(0)
next x
```

```
info_message@("Children is done")

endmacro

' children is the child task called by the NewTasks macro

macro children
var x

for x = 1 to 100
    WORK_IN_PROGRESS@(x++"                              ", "Task2 has control...")
/*
*       The DELAY statement below passes control of the execution
*       thread to the NewTasks macro.
*/
    delay@(0)
next x
endmacro
```

PEND_FOR_NEW_TASK@ Example

```
'  This macro shows how to start a task with
'  PEND_FOR_NEW_TASK@. Note that the new task
'  runs to completion before control is returned to the Parent task.

macro NewTasks

var mac, x
mac = "children"
PEND_FOR_NEW_TASK@(mac)

info_message@("Children is done")

endmacro

' children is the child task called by the NewTasks macro

macro children
var x

for x = 1 to 100
    WORK_IN_PROGRESS@(x++"                              ", "Children has control...")
```

```
next x
endmacro
```

DB_CTRL_RIGHT_MOUSE_MENU@ Example

```
'  This macro shows how to create and display menus for
'  ELF dialog box controls, such as edit boxes and entry boxes.
'
INCLUDE "dialog_.am"

macro mouse_test

    VAR   dbox,exit_cond, ctrl_value, done, id
    VAR FORMAT arrayof rminfo@ info, info1, info2,info3
/*
*       When you stuff the rminfo structures, your menu should
*        have as many items as you have structures. This menu
*       will have one item:
*/
    info[0].name = "Run Graphics"
    info[0].macro_name = "GR_APPLICATION_DLG@"
    info[0].active = TRUE
/*
*       This menu will have two items:
*/
    info1[0].name = "Launch Words"
    info1[0].macro_name = "WP_APPLICATION_DLG@"
    info1[0].active = TRUE
    info1[1].name = "Run Macro dbox"
    info1[1].macro_name = "RUN_MACRO_DLG@"
    info1[1].active = False
/*
*       This menu has one item:
*/
    info2[0].name = "Launch ME   "
    info2[0].macro_name = "ME_APPLICATION_DLG@"
    info2[0].active = TRUE
/*
*       This menu has two menu items, the second of which
*       is grayed out:
*/
```

```
    info3[0].name = "Run Macro dbox  "
    info3[0].macro_name = "RUN_MACRO_DLG@"
    info3[0].active = TRUE  ' menu item is active

    info3[1].name = "LAUNCH SS"
    info3[1].macro_name = "SS_APPLICATION_DLG@"
    info3[1].active = FALSE' menu item is grayed out

    dbox = DB_LOAD@("mouse.d")

    DB_CTRL_RIGHT_MOUSE_MENU@(dbox, "Entry Box", info)
    DB_CTRL_RIGHT_MOUSE_MENU@(dbox, "List Box", info1)
    DB_CTRL_RIGHT_MOUSE_MENU@(dbox, "Table", info2)
    DB_CTRL_RIGHT_MOUSE_MENU@(dbox, "Editbox", info3)

    done = FALSE

    WHILE NOT done

        DB_DISPLAY@(dbox)
        IF DB_CANCELLED@(dbox) RETURN
        exit_cond = DB_EXIT_CTRL@(dbox)

        CASE of exit_cond

            CASE "OK"
                done = TRUE

        ENDCASE
    WEND

ENDMACRO
```

DB_CTRL_INSET@ Example

```
INCLUDE "graphic_.am"
INCLUDE "insets_.am"
INCLUDE "app_ids_.am"

macro sample_inset()
    var format ax_inset_object@ inset_object
    var format ax_inset_info@ inset_info
    var format gr_inset_source@ src
```

```
    var format ax_inset@ inset
    var format ax_units@ units
    var format ax_area@ dest
    var dbox, gfx, wpx, pathname

    if FILE_EXISTS@(SYSTEM_DIR@()++"/new_wp.aw")
            pathname = SYSTEM_DIR@()++"/new_wp.aw"
else if
    FILE_EXISTS@(SYSTEM_DIR@()++"/eng/Samples/News.aw")
            pathname = SYSTEM_DIR@()++"/eng/Samples/News.aw" else if
    FILE_EXISTS@(SYSTEM_DIR@()++"/elf/graphic_.am")
            pathname = SYSTEM_DIR@()++"/elf/graphic_.am"

' create a dialog box containing a canvas

    dbox = db_create_dialog@(0, "Sample", 200, 200)
    db_create_ctrl@(dbox, 8, "MyTab", "MyTab", 0, 0, null)
    db_create_ctrl@(dbox, 3, "OK", "OK", 80, 150, null)
    db_ctrl_button_type@(dbox, "OK", 2)
    db_ctrl_line_thickness@(dbox, "MyTab", 0)
    db_ctrl_width@ (dbox, "MyTab", 400)
    db_ctrl_height@(dbox, "MyTab", 400)
    dest.x = 0
    dest.y = 0
    dest.width = 2              ' inches
    dest.height = 2             ' inches

' create a words object to render onto the Tab control

    wpx_create_instance@(wpx, 0)
    wpx_read_file@(wpx, pathname)       ' Use the data from the file
    src = null
    units.dpi = 1000
    units.dpu = 1000
    units.precision = 3
    units.name = "in."
    inset_object.type = APP#WORDS_
    inset_object.uid = wpx
    inset_object.info = NULL
    inset_object.parents = NULL


' bind the object to the canvas (will automatically display with widget)
```

```
    inset_info.units = units
    inset_info.destination = dest
    inset_info.x_zoom = 100
    inset_info.y_zoom = 100
    inset_info.scale_mode = "clip to fit"
    inset_info.widget_colors = true ' null fields resolved at display time
    inset.obj = inset_object
    inset.info = inset_info
    db_ctrl_inset@(dbox, "MyTab", inset)

    ' keep displaying the dbox until cancelled
 repost:
    db_display@(dbox)
    if not db_cancelled@(dbox)
            goto repost
 endmacro
```

IS_ASCII_FILE@ Example

```
'  This macro presents a list of files and directories
'  ELF dialog box controls, such as edit boxes and entry boxes.
'
macro is_file_ascii

VAR        dbox,
    exit_cond,
    ctrl_value,
    done,
    list_of_dirs,
    home_dir,
    directory,
    lb_string_array,
    index,index1,
    File_string_array,
    File_List,
    pathname,
    result

'load specified dialog box into memory

dbox = DB_LOAD@("isascii.d")
```

```
'initialization section for widgets before being displayed

home_dir = user_dir@()

list_of_dirs = List_of_Dirs@(home_dir)

DB_CTRL_RETURN_ON_CHANGE@(dbox, "List Box", TRUE)
DB_CTRL_RETURN_ON_CHANGE@(dbox, "List Box.1", TRUE)
DB_CTRL_STRINGS@(dbox, "List Box", list_of_dirs)

done = FALSE

WHILE NOT done

'display dialog box to screen

DB_DISPLAY@(dbox)

'get the (optional id) for widget causing exit condition

exit_cond = DB_EXIT_CTRL@(dbox)

'process accordingly based on exit condition


CASE of exit_cond

    CASE "List Box"
    DB_CTRL_TITLE@(dbox, "Label1", "Click a File Name")
    index = DB_CTRL_GET_VALUE@(dbox, "List Box")
    lb_string_array = DB_CTRL_GET_STRINGS@(dbox, "List Box")
    pathname = home_dir++"/"++lb_string_array[index]
    File_List = LIST_OF_FILES@(pathname)

    DB_CTRL_STRINGS@(dbox, "List Box.1", File_List)

    CASE "List Box.1"
    index1 = DB_CTRL_GET_VALUE@(dbox, "List Box.1")
    file_string_array = DB_CTRL_GET_STRINGS@(dbox, "List Box.1")

result = IS_ASCII_FILE@(pathname++"/"++file_string_array[index1])

    If result = -1
            DB_CTRL_TITLE@(dbox, "Label1", "YES - that is an ASCII file")
```

```
    If result = 0
        DB_CTRL_TITLE@(dbox, "Label1", "NO - that is not an ASCII file")

    CASE "OK"
        RETURN

ENDCASE

WEND

ENDMACRO
```