CX-Supervisor
Automatización OLE

ESTE MANUAL CONTIENE:

1.- Introducción

OLE Automatización es una tecnología desarrollada por Microsoft para facilitar el intercambio de información y el control de funciones entre dos programas. Principalmente se trata de un método de programación que permite a un desarrollador de software utilizar las funciones o recursos de otro programa sin tener que programarlas personalmente.

Su aplicación en CX-Supervisor es muy amplia, pues permite utilizar las funcionalidades que poseen otros programas como Excel, Word, etc. Debido a esto, es la tecnología adecuada para intercambiar datos entre CX-Supervisor y cualquier otro programa que admita esta tecnología, siendo el sustituto natural de la obsoleta DDE.

2.- Contexto de Trabajo

Para ilustrar los diferentes capítulos, se van a utilizar CX-Supervisor y Microsoft Excel, el cual, al igual que todos los productos que conforman la suite Office, soporta OLE Automatización.

No es necesario conocer en profundidad ninguno de los dos productos, sin embargo si que es recomendable tener algunas nociones básicas sobre su funcionamiento.

Asimismo, es recomendable conocer algún lenguaje de programación orientado a objetos para entender correctamente el concepto de OLE Automatización y extender el uso de esta tecnología más allá de los ejemplos que se muestran.

No obstante, el uso de esta tecnología es muy sencillo y para seguir los ejemplos que se muestran en Visual Basic (VBScript en Supervisor) no es necesario haber programado antes en este lenguaje.

3.- Metodología

3.1 El objeto SYSMAC.SCS.POINT

El uso de OLE Automatización se basa en objetos, de ahí la recomendación de conocer algún lenguaje de programación orientado a objetos.

Un objeto no es más que una agrupación de variables y/o funciones que el desarrollador del software ha puesto a disposición del usuario. Mediante el uso de estos objetos y las funciones y variables que contiene, una aplicación podrá interactuar o intercambiar datos con otras.

Antes de poder usar los objetos, hay que crearlos. Este paso se realiza en Visual Basic (y VBScript) de la siguiente manera:

```
Set MyObj = CreateObject("ProgId.ObjectName")
```

Aunque en Visual Basic no es necesario, es muy recomendable definir las variables antes de usarlas, por lo que el código quedaría así:

```
Dim MyObj As Object
Set MyObj = CreateObject("ProgId.ObjectName")
```

Si el lenguaje de programación fuera VBScript, el código quedaría así::

```
Dim MyObj
Set MyObj = CreateObject("ProgId.ObjectName")
```

Una vez creado el objeto ya se puede utilizar siguiendo la metodología estándar de Visual Basic:

```
MyObj.InvokeSomeMethod
```

El objeto público de CX-Supervisor es POINT, y se debe invocar a través de SYSMAC.SCS. Además, el objeto POINT requiere de un parámetro adicional que le indicara el tipo de puntos al que se va a acceder. Es parámetro puede ser 1 ó 2. Con esto quedaría:

```
Dim scs As Object
Set scs = CreateObject("SYSMAC.SCS.POINT.1")
```

3.2 Uso de Arrays

En el caso de usar arrays, el parámetro que se debe pasar al objeto debe ser 2. Si por el contrario no se van a utilizar, se debe usar el parámetro 1.

```
Set scs = CreateObject("SYSMAC.SCS.POINT.2")
```

La forma de crear una variable para almacenar todo el contenido de un Array de CX-Supervisor en Visual Basic se debe hacer usando la función Redim, por ejemplo:

```
Dim vArray As Variant
ReDim vArray(10) As Double
```

3.3 Tipos de variables en Visual Basic

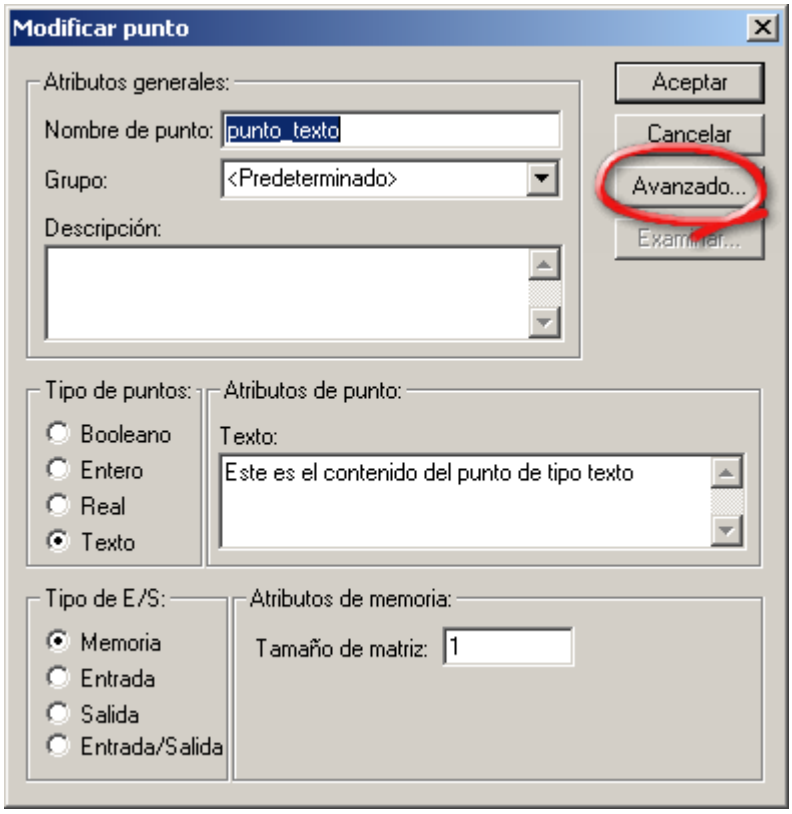
Los tipos de puntos de CX-Supervisor no se corresponden con los tipos de variables que permite Visual Basic, por lo que a la hora de crear una determinada variable para almacenar el valor de un punto de CX-Supervisor hay que prestar atención al siguiente cuadro:

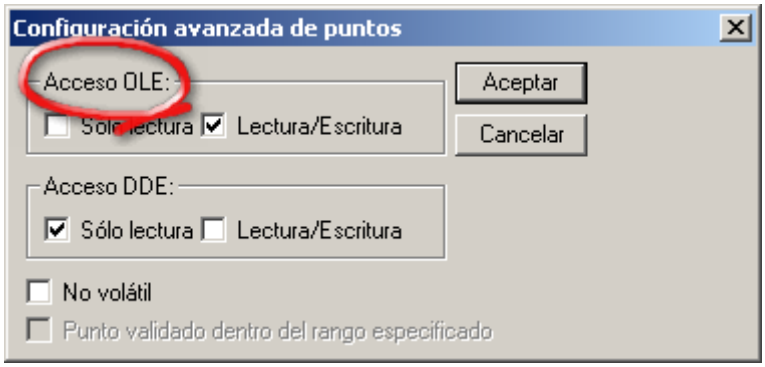
Tipo de punto en CX-Supervisor	Variable de Visual Basic
`Booleano`	`Integer`
`Entero`	`Long`
`Real`	`Double`
`Texto`	`Text`

3.4 Permisos de lectura y escritura OLE

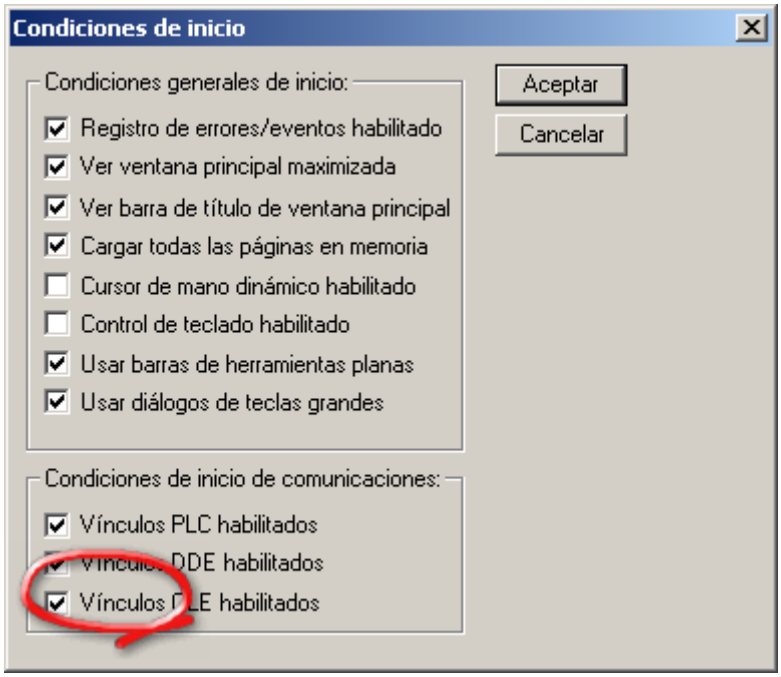
Para que valor de los puntos de CX-Supervisor pueda ser leído y/o modificado, los puntos deben tener los permisos adecuados.

Para acceder a la configuración OLE de un punto, basta con pulsar sobre el botón de “Avanzado” en la ventana de edición de puntos:





En este cuadro de diálogo se elegirá si el valor del punto se puede leer, escribir o ninguna de las dos cosas.
Además, para que el intercambio de datos funcione, las comunicaciones OLE deben estar habilitadas en la aplicación. Para hacer esto, basta con ir a Proyecto → Configuración de la Aplicación → Condiciones de Inicio y marcar la opción de habilitar Vínculos OLE.



Además de esta opción, las comunicaciones OLE se pueden habilitar o deshabilitar en cualquier momento a través del comando de script **`EnableOLE (vBoolean As Boolean)`**, donde `vBoolean` puede tomar los valores TRUE o FALSE según se desee habilitar o deshabilitarlas.

3.5 Eliminación de objetos y liberación de memoria

Cuando se crea un objeto se ocupa una determinada cantidad de memoria y recursos del ordenador. Cuando ese objeto se termina de utilizar, se debe eliminar de la memoria para que esa memoria y los recursos que antes estaban en uso, ahora estén disponibles para otra aplicación.

Para eliminar la conexión existente entre la aplicación cliente y la servidora (la conexión con el objeto), se utiliza el método `Quit`[1]:

```
MyObj.Quit
```

Y para liberar la memoria ocupada por ese objeto, hay que aplicar la referencia `nothing` a la variable que contiene la referencia al objeto:

```
Set MyObj = Nothing
```

En ocasiones no es necesario realizar este paso, pues también la aplicación servidora (objeto creado) puede limitar sus instancias a una, puede cerrarse mediante los métodos convencionales (pulsando sobre el aspa de su

[1] La aplicación OLE Servidora debe soportar este método.

ventana) o por que el objeto y su referencia desparecen al terminar su ámbito de visibilidad.

No obstante, hay que tener en cuenta que estos pasos son importantísimos, pues si el objeto y su referencia no desapareciera, cada nuevo objeto que se creara permanecería en memoria ocupando una parte del total disponible hasta que el ordenador quedara bloqueado por falta de recursos.

4.- Integración de Aplicaciones

4.1 Intercambio de Datos

Cuando se habla de intercambio de datos se hace referencia sencillamente a la lectura y escritura de datos de un programa a otro.

CX-Supervisor nos proporciona las funciones necesarias para poder leer y escribir datos en él, es decir conocer o modificar el valor de sus puntos.

Estas funciones son:

- **GetValue()** permite monitorizar cualquier punto que tenga habilitado el acceso OLE para lectura
- **GetArray()** igual que el anterior, pero permite monitorizar los puntos definidos como Array
- **SetValue ()** permite modificar cualquier punto de CX-Supervisor que tenga habilitado el acceso OLE de escritura
- **SetArray()** igual que el anterior, pero permite modificar los puntos definidos como Array

Ejemplo 1: Leer y escribir puntos de CX-Supervisor desde Excel

Requerimientos: Antes de continuar se ha de crear una aplicación en CX-Supervisor que incluya los puntos **punto_booleano**, **punto_entero**, **punto_texto** y **punto_array** definidos como booleano, entero, texto y entero de 10 elementos respectivamente. Además, sus permisos OLE deben ser de lectura/escritura. Se puede crear también una página que muestre el valor de esos puntos y desde la que se puedan modificar.

Pasos:

- Abrir Excel
- Pasar a modo de diseño, para ello simplemente pulsar sobre el botón  . Si no apareciese este icono, ir a Ver → Barras de Herramientas → Cuadro de Controles
- Introducir dos botones en la hoja de Excel  , dejando sin tapar la primera columna de celdas. El primer botón servirá para recoger el valor de los puntos de CX-Supervisor, el segundo para modificar esos datos.
-  Hacer doble clic sobre el primero de los botones. Al hacer esto, aparecerá el editor de Visual Basic.
- Se introduce el siguiente código:

```
    Dim resultado As Integer

    ' Definimos el objeto scs
    Dim scs As Object

    ' Creamos la variable para el array recogido de Supervisor
    Dim vArray As Variant
    ReDim vArray(10) As Double

    ' Creamos el objeto -conectamos- CX-Supervisor
    Set scs = CreateObject("SYSMAC.SCS.POINT.2")

    ' Recogemos el valor de los puntos y los mostramos en la
primera columna
    Cells(2, 1) = scs.GetValue("punto_booleano")
    Cells(3, 1) = scs.GetValue("punto_entero")
    Cells(4, 1) = scs.GetValue("punto_texto")

    ' Recogemos el valor del punto definido como Array y lo
mostramos en la primera columna
    resultado = scs.GetArray("punto_array", vArray)
```

```
    For n = 6 To 15
        Cells(n, 1) = vArray(n - 6)
    Next n
```

- 

Hacer doble clic sobre el segundo botón
- Se introduce el siguiente código:

```
    Dim resultado As Integer

    ' Definimos el objeto scs
    Dim scs As Object

    ' Creamos la variable para el array a escribir en Supervisor
    Dim vArray As Variant
    ReDim vArray(10) As Double

    ' Creamos el objeto -conectamos- CX-Supervisor
    Set scs = CreateObject("SYSMAC.SCS.POINT.2")

    ' Recogemos el valor de los puntos de las celdas y los
escribimos en los puntos de Supervisor
    resultado = scs.SetValue("punto_booleano", Cells(2, 1))
    resultado = scs.SetValue("punto_entero", Cells(3, 1))
    resultado = scs.SetValue("punto_texto", Cells(4, 1))

    ' Recogemos el de las celdas, lo introducimos en un Array y
lo escribimos en Supervisor
    For n = 6 To 15
        vArray(n - 6) = Cells(n, 1)
    Next n
    resultado = scs.SetArray("punto_array", vArray)
```

- Cerrar el editor de Visual Basic
- Guardar el libro de Excel
- Pasar a modo Runtime, para ello pulsar de nuevo sobre 

Al pulsar el primer botón, en la primera columna del libro de Excel se escribirán los valores de los puntos de CX-Supervisor. De la misma manera, al pulsar sobre el segundo botón, los datos que se encuentren en la primera columna de Excel se escribirán en CX-Supervisor.

4.2 Funciones Adicionales

Además de las funciones comentadas en el apartado anterior, CX-Supervisor pone a disposición del programador otra serie de funciones:

- **QueryCount()** devuelve el número total de puntos de la base de datos de CX-Supervisor
- **QueryId()** retorna la ID de un punto (en formato WORD)
- **QueryOLE()** Devuelve los derechos de lectura/escritura de un punto
- **QueryType()** Devuelve el tipo de punto, por ejemplo: Booleano, Entero, Real, etc.
- **QueryName()** devuelve el nombre de un punto, pasado como un identidad, como cadena

A través de estas funciones se puede crear un listado de puntos de la aplicación de Supervisor. Esto es útil, puesto que no siempre se conoce el nombre de un punto específico para pasarlo a las funciones de intercambio de datos.

Ejemplo 2: Recoger el listado de puntos de CX-Supervisor

Pasos:

- Abrir Excel
- Pasar a modo de diseño, para ello simplemente pulsar sobre el botón  . Si no apareciese este icono, ir a Ver → Barras de Herramientas → Cuadro de Controles
- Introducir dos botones  y cuatro cuadros combinados  en la hoja de Excel. El primer botón servirá para conectar con CX-Supervisor y recoger el listado de puntos, que se almacenarán en los cuadros combinados. El segundo botón permitirá borrar el contenido de esos cuadros combinados.
- Dar los siguiente nombres, a través de la página de propiedades, a los cuadros combinados:

  - cboDigitalName
  - cboIntegerName
  - cboRealName
  - cboTextName
  - cboSystemName

-  Hacer doble clic sobre el primero de los botones. Al hacer esto, aparecerá el editor de Visual Basic. Introducir el siguiente código:

```
Dim nNumberOfPoints, nIndex, nType As Integer
Dim gssIOOLE As Object
Dim strName As String

Set gssIOOLE = CreateObject("SYSMAC.SCS.POINT.1")

'Recogemos el número de puntos en Supervisor
nNumberOfPoints = gssIOOLE.QueryCount()

For nIndex = 1 To nNumberOfPoints
    'Recogemos sus nombres
    strName = gssIOOLE.QueryName(CInt(nIndex))
    'Recogemos el tipo de cada datos
    nType = gssIOOLE.QueryType(CInt(nIndex))
    'Comprobamos los puntos de sistema
    nPos = InStr(strName, "$")
    If nPos = 0 Then
        If nType = 1 Then
            'Digital Point
            cboDigitalName.AddItem strName
        Elseif nType = 2 Then
            'Integer Point
            cboIntegerName.AddItem strName
        Elseif nType = 3 Then
            'Real Point
            cboRealName.AddItem strName
        Elseif nType = 4 Then
            'Text Point
            cboTextName.AddItem strName
```

```
            End If
        Else
            cboSystemName.AddItem strName
        End If
        cboDigitalName.ListIndex = cboDigitalName.ListCount - 1
        cboIntegerName.ListIndex = cboIntegerName.ListCount - 1
        cboRealName.ListIndex = cboRealName.ListCount - 1
        cboTextName.ListIndex = cboTextName.ListCount - 1
        cboSystemName.ListIndex = cboSystemName.ListCount - 1
    Next
```

-  Hacer doble clic en el segundo botón e introducir el siguiente código:

```
    For i = 0 To cboDigitalName.ListCount - 1
        cboDigitalName.RemoveItem (0)
        cboDigitalName.Value = ""
    Next
    For i = 0 To cboIntegerName.ListCount - 1
        cboIntegerName.RemoveItem (0)
        cboIntegerName.Value = ""
    Next
    For i = 0 To cboRealName.ListCount - 1
        cboRealName.RemoveItem (0)
        cboRealName.Value = ""
    Next
    For i = 0 To cboTextName.ListCount - 1
        cboTextName.RemoveItem (0)
        cboTextName.Value = ""
    Next
    For i = 0 To cboSystemName.ListCount - 1
        cboSystemName.RemoveItem (0)
        cboSystemName.Value = ""
    Next
```

- Cerrar el editor de Visual Basic
- Guardar el libro de Excel
- Pasar a modo Runtime, para ello pulsar de nuevo sobre 

Ahora, al pulsar sobre el primero de los botones, los puntos de la aplicación de Supervisor aparecerán en los cuadros combinados de acuerdo con el tipo (booleano, entero, etc.)
Al pulsar el segundo botón, se borrará el contenido de todos los cuadros combinados.

4.2 Controlando otras aplicaciones desde CX-Supervisor

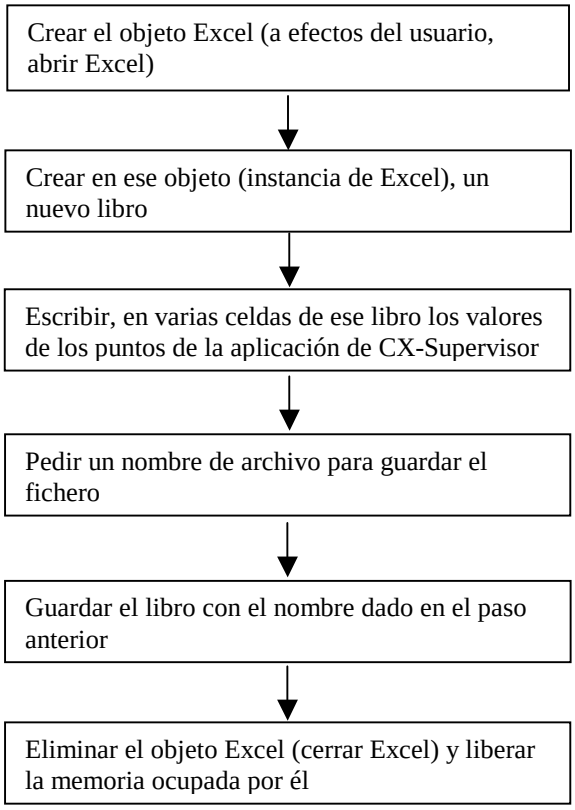
Existen aplicaciones tales como Excel, Word, etc. que además de presentar funciones para el intercambio de datos, también presentan otras muchas que permiten controlar totalmente la aplicación.

En el ejemplo 1, el código escrito en Excel, permite realizar un intercambio de datos con CX-Supervisor, pero de la misma forma, un código en CX-Supervisor puede realizar un intercambio de datos con Excel.

Además, Excel, presenta un gran número de funciones que permiten controlarlo totalmente usando OLE Automatización.

En el siguiente ejemplo, mediante código VBScript, CX-Supervisor creará un objeto Excel, un libro nuevo, rellenará unas columnas con varios valores de puntos y finalmente se pedirá un nombre de archivo para guardar ese libro.

Es decir, el script de VB en CX-Supervisor debe permitir realizar lo siguiente:



Teniendo esto en cuenta ya se puede comenzar con el ejemplo.

Ejemplo 3: Controlar Excel desde CX-Supervisor

Requerimientos: Antes de continuar se ha de crear una aplicación en CX-Supervisor que incluya los puntos **punto_booleano**, **punto_entero**, **punto_texto** y **punto_array** definidos como booleano, entero, texto y entero de 10 elementos respectivamente. También se debe crear una página desde la cual se pueda modificar el valor de los puntos y donde además se encuentre un botón para ejecutar el script de VB.
También se debe crear una carpeta de nombre "`Ficheros Generados`" dentro de la carpeta que contiene el proyecto de CX-Supervisor.

Pasos:

-  Añadir un script al botón, de tipo VBScript.
- Introducir el siguiente código:

```
' Declaración de las variables a usar

Option Explicit
```

```
Dim objExcel, n, fileName

' 1.- Crear el objeto Excel (a efectos del usuario, abrir Excel)

Set objExcel = CreateObject ("Excel.Application")
objExcel.Visible = True


' 2.- Crear un nuevo libro

objExcel.Workbooks.Add

' 3.- Escribir, en varias celdas de ese libro los valores de los puntos
de la aplicación de CX-Supervisor

objExcel.Cells(1, 1) = "Valores de los puntos de CX-Supervisor"
objExcel.Cells(2, 1) = punto_booleano
objExcel.Cells(3, 1) = punto_entero
objExcel.Cells(4, 1) = punto_texto
objExcel.Cells(5, 1) = "Valores de los elementos del array"
For n = 6 To 15
       objExcel.Cells(n, 1) = punto_array (CInt (n - 6))
Next

' 4.- Pedir un nombre de archivo para guardar el fichero

fileName = InputBox ("Introduzca un nombre para el fichero (sin la
extensión)...", "Selección de fichero")

' 5.- Guardar el libro con el nombre dado en el paso anterior
' s_ProjectPath es un punto de sistema que indica el path del proyecto

objExcel.ActiveWorkbook.SaveAs s_ProjectPath & "\Ficheros Generados\" &
fileName & ".xls"

' 6.- Eliminar el objeto Excel (cerrar Excel) y liberar la memoria
ocupada por él

objExcel.Quit
Set objExcel = Nothing
```

Al ejecutar este código[2], automáticamente se creará el libro de Excel, se rellenarán las celdas con los valores de los puntos de Supervisor y se guardará el fichero de Excel con el nombre dado dentro de la carpeta "`Ficheros Generados`" que cuelga de la carpeta de proyecto de Supervisor.

En el ejemplo, al crear el objeto `Excel.Application`, Excel aparece de la misma manera que cuando se ejecuta directamente, es decir, es visible para el usuario. Sin embargo, también es posible que Excel no sea visible, siendo todo el proceso todavía más transparente para el usuario. Para hacer esto, simplemente habría que sustituir la línea `objExcel.Visible = True` por `objExcel.Visible = False`.

[2] Nota: El script mostrado para CX-Supervisor es válido únicamente para las versiones 1.2 y superiores.