Sumario

INTRODUCCIÓN 6
Agradecimientos 6
COMENZAMOS 7
Pestaña Form: 8
Pestaña Coordinates; 9
Pestaña Style; 9
Pestaña StatusBar: 9
Pestaña Resource: 9
Pestañas OLE y Compatibility: 9
Pestaña Colors: 9
Pestaña Fonts: 10
IMÁGENES ESTÁTICAS y/o DINÁMICAS. 11
Dinámica: 11
Estática: 11
Diseño del Form: 13
LOS CONTROLES de PWC. 15
PowerCOBOL CheckBox Control: 15
Pestaña CheckBox: 15
PowerCobol ComboBox Control: 15
Pestaña ComboBox: 16
Pestaña File: 16
PowerCOBOL CommandButton Control , ... 16
Pestaña CommandButton: 16
Pestaña Image: 17
PowerCobol GroupBox Control 17
Pestaña Group: 18
PowerCobol ListBox Control 18
Pestaña ListBox: 18
PowerCobol OptionButton Control. 19
Pestaña Option Button: 19
PowerCobol ScrollBar Control 19
Pestaña ScrollBar: 20
PowerCobol StaticText Control 20
Pestaña StaticText: 20
Pestaña RenderTexts: 20
Pestaña Appearence: 20
PowerCobol TextBox Control 21
Pestaña TextBox: 21
Pestaña SingleLine: 21
Pestaña Multiline: 22
Pestaña RenderTexts: 22
PowerCobol Frame Control 23
Pestaña frame: 23
PowerCobol ListView Control 23
Pestaña ListView: 24
Pestaña Column: 24
Pestaña RenderText: 24
PowerCobol ProgressIndicator Control 24
Pestaña ProgressIndicator: 25
PowerCobol Slider Control 25
Pestaña Slider1: 25

Pestaña Slider2: 26
PowerCobol TreeView Control 26
Pestaña TreeView: 26
PowerCobol Tab Control. 27
Pestaña Tab: 27
Pestaña Page: 27
PowerCobol ToolBar Control. 28
Pestaña Toolbar: 28
Pestaña Button: 28
PowerCobol DriverList Control. 29
Pestaña DriverList: 29
PowerCobol FileList Control. 29
Pestaña FileList: 30
PowerCobol FolderList Control. 30
Pestaña FolderList: 30
PowerCobol Timer Control. 30
Pestaña Timer: 31
PowerCobol Shape Control. 31
Pestaña Shape: 31
PowerCobol Graph Control. 31
Pestaña Graph: 32
PowerCobol Table Control. 32
Pestaña Table: 32
Pestaña Rows: 33
Pestaña Columns: 33
PowerCobol Animation Control. 33
Pestaña Animation: 33
PowerCobol Image Control. 34
Pestaña Image: 34
Pestaña Extend: 34
PowerCobol MCI Control. 35
Pestaña MCI: 35
PowerCobol DB Access Control. 35
PowerCobol DDE Control. 36
PowerCobol Print Control. 36
PowerCobol ADO DataSource Control. 37
Pestaña ADO DataSoure: 37
PowerCobol Label Control. 37
PowerCobol Edit Control. 37
Pestañas comunes a todos los controles >> 38
Pestaña Colors: 38
Pestaña Coordinates: 38
Pestaña Appearance: 38
Pestaña Common: (ésto es aplicable a todos los controles) 38
<< Otras acciones >> 39
Crear un Array: 40
Vamos con la BASE de DATOS. 42
Crear la BD 42
ODBC: 43
.INF: 43
COBOL85.cbr: 43
Y por fin … vamos con la programación. 45
<< Consideraciones antes de empezar >> 45
Nuestro programa: 47
Programación del Form 48
ENVIRONMENT DIVISION → SPECIAL-NAMES: 48

ENVIRONMENT DIVISION → FILE CONTROL:.....48
DATA DIVISION → FILE SECTION:.....48
DATA DIVISION → WORKING-STORAGE SECTION:.....48
Eventos del Form.....50
Evento *Closed* del form:.....50
Evento *Opened* del form:.....50
Programación de los Controles.....51
Evento *keypress* del campo1:.....51
Evento *Return* del campo1.....52
Evento *KeyPress* de los campos 2, 3, 4, 9, 10 y 11.....53
Evento KeyPress de los campos 5, 6, 7 y 8.....53
Evento *Return* de los campos 2, 4, 5, 6, 7, 8, 9 y 10.....54
Evento *KeyPress* del campo3.....54
Evento *Return* del campo3.....55
Eventos en un Array de Controles.....55
Evento *DblClick* en Imagen1_n:.....56
Evento *Click* en BtImg1_n:.....56
Evento *Click* en BtImg2_n:.....57
Evento *ButtonClick* de CmToolbar1:.....57
Evento *Click* del botón BTGRABAR:.....58
Evento *Click* del botón BTCANCEL:.....60
Evento *Click* del botón BTCONSUL:.....60
Evento *Click* del botón BTBORRAR:.....61
Evento *Click* del botón BTIMPRIMIR:.....63
Evento *Click* del botón BTLISTAR:.....63
Evento *Click* del botón BTSALIR:.....63
Evento *Click* del botón BTB1:.....64
Evento *Click* de los botones BTB1 y BTB2:.....65
Evento *Click* del botón BTB2:.....65
Otros eventos de los Controles.....66
Subrutinas.....67
Subrutina “EXISTE”.....68
Subrutina “HABILITAR”.....69
Subrutina “INHABILITAR”.....69
Subrutina “MOSTRAR-FAM”.....70
Subrutina “NO-EXISTE”.....70
Subrutina “PARA-SALIR”.....71
Subrutina “VACIAR”.....71
Variables propias de PWC.....72
PWC y la API de Windows. ALPHAL(WORD).....88
El Programa “*CONSULTAS*”.....89
Programación del Form (Consultas).....90
ENVIRONMENT DIVISION → SPECIAL-NAMES:.....90
ENVIRONMENT DIVISION → REPOSITORY.....90
ENVIRONMENT DIVISION → FILE CONTROL:.....90
DATA DIVISION → FILE SECTION:.....90
DATA DIVISION → WORKING-STORAGE SECTION:.....90
Evento KeyDown del form:.....97
Evento Opened:.....97
Evento *SelChange* de COMBO2:.....98
Evento *KeyPress* de CAMPO1:.....100
Evento *Return* de CAMPO1:.....100
Evento *KeyPress* de CAMPO2:.....100
Evento *Return* de CAMPO2:.....100
Evento *ColumnClick* de VIEW2:.....101
Evento *DblClick* de VIEW2:.....101

Evento *KeyPresss* de VIEW2: 101
Evento *Click* de BTEXCEL: 102
Evento *Click* de CHECK4: 102
Evento *Click* de botón BTRECARGAR: 102
Evento *Click* de botón BTLISTAR: 102
Evento *Click* de botón BTSALIR 104
Evento *Click of GRUPO-PRN* de botón BTPRN-SI: 104
Evento *Click of GRUPO-PRN* de botón BTPRN-NO: 106
Evento *Click of GRUPO-PRN* de botón BTPRN-SI: 106
Subrutinas programa CONSULTAS 107
Subrutina “A-EXCEL” 107
Subrutina “CARGAR” 110
Subrutina “CARGAR1” 113
Subrutina “CARGAR2” 116
Subrutina “CARGAR-N” 119
Subrutina “CARGAR-N” 125
Subrutina “PARA-SALIR” 129
El Programa “*LISTADO*” 130
Programación del Form (Listado) 131
ENVIRONMENT DIVISION → SPECIAL-NAMES: 131
ENVIRONMENT DIVISION → REPOSITORY: 131
ENVIRONMENT DIVISION → FILE CONTROL: 131
DATA DIVISION → FILE SECTION: 131
DATA DIVISION → WORKING-STORAGE SECTION: 131
Eventos del Form 134
Evento *Opened* del form: 134
Eventos de los controles del Form 135
Evento *Return* de campo1: 135
Evento *Click* de botón BTCOD1: 135
Evento *Return* de CAMPO2: 151
Evento *Click* de botón BTCOD2: 151
Evento *Return* de CAMPO3: 152
Evento *Click* de botón BTCOD3: 152
Evento *Return* de CAMPO4: 153
Evento *Click* de botón BTCOD4: 154
Evento *Click* de *CmCheck* CHECK5: 154
Evento *Click* de *CmCheck* CHECK6: 155
Evento *Return* de CAMPO11: 155
Evento *Click* de botón BTFIC: 155
Evento *Return* de CAMPO12: 156
Evento *Click* de *CmCheck* CHECK7: 156
Evento *Click* de botón BTIMP: 157
Evento *Click* de botón BTLISTAR: 157
Evento *Click* de botón BTSALIR: 164
Subrutinas del programa “Listados” 165
Subrutina “A-EXCEL” 165
Subrutina “CARGAR” 168
Subrutina “POR-IMPRESORA” 170
Subrutina “VACIAR” 176
Consideraciones: 177
Rutinas CBL_xxxxx/CBL_xxxxx2/CBL_64BIT_xxxx: 177
CBL_OPEN_FILE/CBL_OPEN_FILE2/CBL_OPEN_64BIT_FILE 188
CBL_CREATE_FILE/CBL_CREATE_FILE2/CBL_CREATE_64BIT_FILE 188
CBL_CLOSE_FILE/CBL_CLOSE_64BIT_FILE 189
CBL_FLUSH_FILE/CBL_FLUSH_64BIT_FILE 190
CBL_CHANGE_DIR/CBL_CHANGE_DIR2 191

CBL_CHECK_FILE_EXIST/CBL_CHECK_FILE_EXIST2 192
CBL_COPY_FILE/CBL_COPY_FILE2 193
CBL_CREATE_DIR/CBL_CREATE_DIR2 193
CBL_DELETE_DIR/CBL_DELETE_DIR2 194
CBL_DELETE_FILE/CBL_DELETE_FILE2 194
CBL_READ_DIR 195
CBL_RENAME_FILE/CBL_RENAME_FILE2 195
CBL_TOUPPER/CBL_TOLOWER 196
CBL_ALARM_SOUND 197
Otras consideraciones. 197
FUNCIONES INTRÍNSECAS. 198
FUNCIÓN ADDR 199
FUNCIÓN LENG/LENGTH 199
FUNCIÓN STORED-CHAR-LENGTH 199
FUNCIÓN CURRENT-DATE 204
FUNCIÓN SUM 205
FUNCIÓN MEDIAN 205
FUNCIÓN MEAN/MIDRANGE 206
FUNCIÓN REM 206
FUNCIÓN MAX/MIN 206
FUNCIÓN INTEGER-OF-DATE/DATE-OF-INTEGER 207
FUNCIÓN SIN/COS/TAN 208
FUNCIÓN ASIN/ACOS/ATAN 208
FUNCIÓN FACTORIAL 208
FUNCIÓN INTEGER/INTEGER-PART 209
FUNCIÓN INTEGER-OF-DAY 209
FUNCIÓN LOG/LOG10 209
FUNCIÓN MOD 209
FUNCIÓN NUMVAL/NUMVAL-C 210
FUNCIÓN ORD 210
FUNCIÓN ORD-MAX/ORD-MIN 210
FUNCIÓN RANDOM 211
FUNCIÓN RANGE 212
FUNCIÓN SQRT 212
FUNCIÓN CHAR 213
FUNCIÓN LOWER-CASE/UPPER-CASE 213
FUNCIÓN REVERSE 213
FUNCIÓN WHEN-COMPILED 213
Se acabó y Contacto … 215
Créditos: 216
Diario de cambios (o changelog para los modernos) 217

INTRODUCCIÓN

Fujitsu Power-Cobol ® , en adelante **PWC**, es un entorno de programación IDE, (**I**ntegrated **D**evelopment **E**nvironment o *Entorno de Desarrollo Integrado*), basado en el lenguaje de programación **COBOL**, del que toma todas sus ventajas, bondades, potencia y facilidad de desarrollo, para crear un entorno de programación sencillo y estructurado.

Éste pequeño manual, intentará ayudarte a dar tus primeros pasos en PWC, creando un proyecto desde cero, hasta llegar a tener un ejecutable del programa totalmente funcional. Para ello vamos a utilizar la versión 9.0 del compilador, aunque sirve igualmente para las versiones 7.n y 8.n, pues los cambios entre estas versiones fueron mínimas. (Aunque, supongo que valdría desde la versión 5.n hasta la 9.n, para la versión 3.n, es seguro que no). Hay que tener en cuenta que no es un manual para aprender a programar en COBOL, por lo que se ha de presuponer que el lector, tiene unos conocimientos mínimos de éste lenguaje.

En éste manual utilizaremos tanto ficheros nativos de COBOL, como enlaces a Bases de Datos, en este caso utilizaremos MariaDB ®, (https://mariadb.org/), que es una base de datos GPL Libre, derivada de MySql ®, con una gran potencia en el manejo de datos y, sobe todo con una ingente cantidad de información en Internet y, como visor/gestor, (o Front-End, como le gusta a los *modernos*, llamarlo ahora), de la BD, Tablas, etc, utilizaremos HeidiSql, (https://www.heidisql.com/), que también es con licencia GPL Libre y cuenta con mucha documentación en las redes. También quiero hacer notar que, éste manual ha sido escrito con *Writer* ® el software de escritura de la suite gratuita LibreOffice, y exportado a Microsoft ® Word 2007 ®, y, aunque las exportaciones, LibreOffice las hace correctamente, puede ser que el manual no se vea del todo tan bien como se ve el original en *Writer* de LibreOffice.

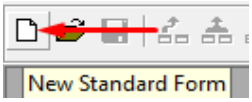
Agradecimientos

Antes de empezar con el proyecto, no estaría bien que no mencionara mi más eterna gratitud al foro de COBOL en español al que estoy suscrito, (https://www.cobolforo.es/index.php), y del que soy uno de los admins, por toda la ayuda prestada en mis muchas consultas, dudas, etc. En éste manual, algunas de las rutinas que ponga, han salido de personal de ese foro, así que **mil gracias** a **Kuk,** (gran jefe del foro :-) **), Rui, hrmcobol, Fito, Roger**, etc, pero en especial, con muchísimo afecto y cariño a **RAPINTO**, que nos dejó hace poco y cuya pérdida, todos los coboleros sentimos en el alma, (fue uno de los mejores en COBOL, pero aún fue mejor persona), y mil perdones por los que me he olvidado, que no ha sido a propósito.

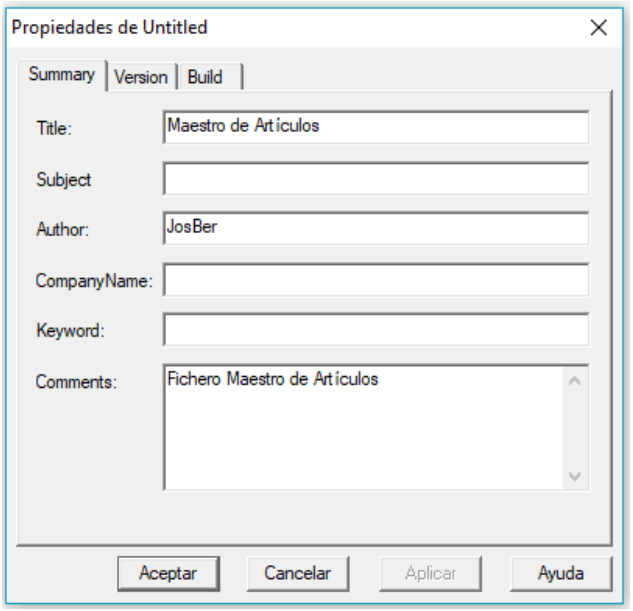
Vamos a hacer un proyecto en el que crearemos un fichero de artículos, que será en SQL y un fichero de familias, que será nativo de COBOL. Crearemos un ALTA/BAJA/MODIFICACIÓN en una misma pantalla, otra pantalla de CONSULTAS y uno o varios LISTADOS. Como editor de código fuente, vamos a utilizar el propio que trae incorporado el PWC por su facilidad de uso, y para crear los listados, vamos a utilizar PowerForm ®, que aunque es un poco espartano y “pesado” de utilizar, funciona muy bien y cumple perfectamente su cometido, así que… vamos allá.

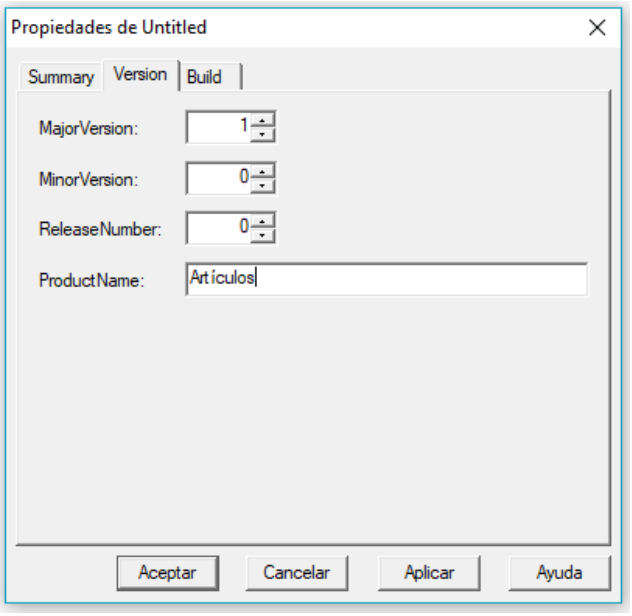
COMENZAMOS

Lo primero es abrir PWC, por lo que vamos a inicio y buscamos Fujitsu NetCobol V9, y dentro, encontraremos **PowerCOBOL**, lo abrimos y pulsamos en el icono *New Standard Form*



Con esto, creamos un proyecto en blanco, que va a ser nuestro punto de partida. Lo primero que vamos a hacer es configurar las propiedades del proyecto, para ello, pulsamos con el botón derecho en el elemento del árbol que pone *Untitled [Project]*, y rellenamos los datos de las 2 primeras pestañas a nuestro gusto, teniendo en cuenta que, lo que pongamos aquí, será lo que luego salga en las propiedades al darle con el botón derecho al ejecutable final.





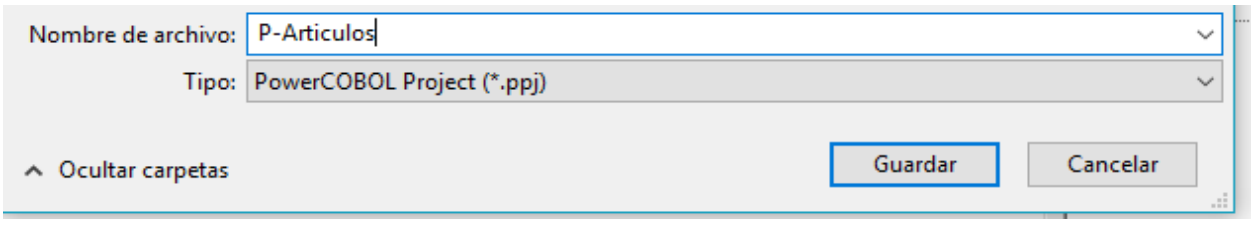
La tercera pestaña, (*Build*), ya es más importante configurarla bien, Yo para el código fuente, he creado una carpeta en el disco duro que se llama ***Manual***, donde guardaremos los ficheros fuente, librearías, iconos, etc y para los ejecutables, otra que se llama ***Articulos***. En ésta tercera pestaña, es donde le vamos a indicar al compilador que los ejecutables se van a guardar en la carpeta ***Articulos***. De momento, de esta pestaña, sólo nos interesan 2 campos, que son:

Debug Mode y *Release Mode*

Debug Mode es la carpeta que utilizará el compilador cuando tengamos que utilizar el *Debugger* de PWC
Release Mode es la carpeta que utilizará el compilador para guardar el ejecutable del programa.

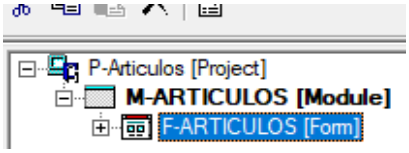
Yo en mi caso, utilizo, (y así lo recomiendo), la misma carpeta en ambos casos.

Pulsamos en **Aceptar**, y ya tenemos configuradas las propiedades de nuestro proyecto. Por lo que vamos a guardarlo, pulsamos en el icono de la carpeta y, cómo es la primera vez que lo guardamos y aún no le hemos dado un nombre, aprovechamos para dárselo, y recuerda guardarlo en la carpeta **Manual**, que es donde hemos quedado en guardar los fuentes, librerías y demás. Yo suelo grabar el nombre con el sufijo *P-*, por temas de tenerlo luego todo ordenado, (sé que todos los archivos que empiezan así, son fuentes del programa), pero cada uno puede hacerlo como quiera.



En el árbol de proyecto, vemos que tenemos un apartado que pone ***Main [Module]***, eso explicado de forma sencilla es que, cada "*Module*", vendría a ser un conjunto de programas del mismo ámbito, (Artículos, Clientes, etc.), y cada *Form*, el que hay debajo de *Module*, serían los programas que forman ese conjunto, (Altas, Consultas, Listados, etc), yo soy partidario, por limpieza y por evitar problemas de llamadas entre programas, (las llamadas no pueden ser recursivas o circulares), de crear un proyecto para cada cosa, (uno para el maestro de artículos, otro para familias, etc,). Además los Módulos los empiezo con el prefijo *M-* y los Form con el prefijo *F-*, pero insisto, en mi caso es por claridad y limpieza, cada uno puede nombrarlos como quiera.

Por ello, vamos a cambiar el nombre al "Module" y al "Form", para ello, pulsamos con el botón derecho en cada uno de ellos y les cambiamos los nombres por ***M-ARTICULOS*** y **F-ARTICULOS**, con lo que debería quedaros más o menos así

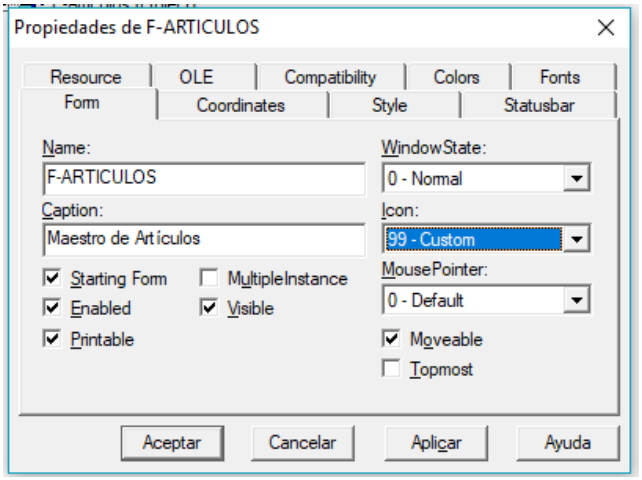


Ahora vamos a configurar el Form, (el Module, de momento no hace falta), por lo que pulsamos con el botón derecho, vamos a *Properties*, y… vamos allá con el montón de pestañas con las que nos encontramos, no hay que preocuparse, porque ésto sólo se suele tocar una vez, y es más rápido y sencillo de lo que parece.

Pestaña Form:

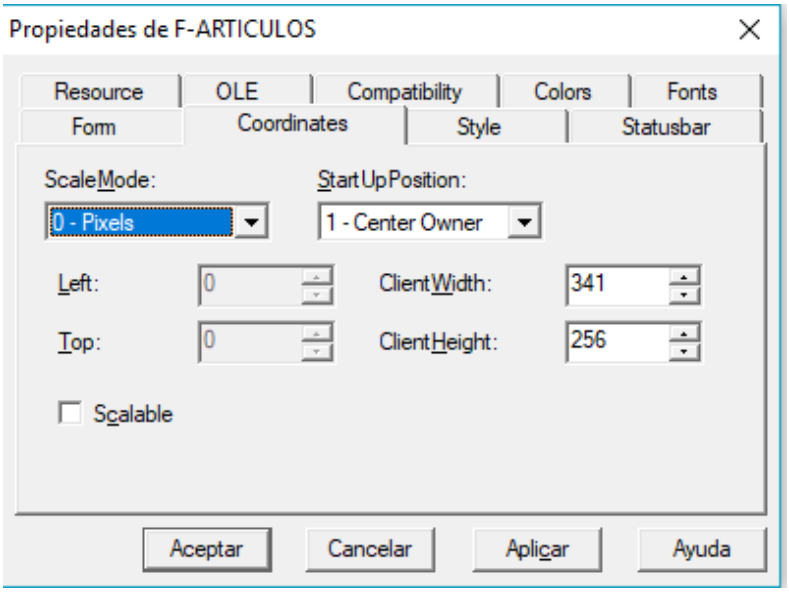
Es de las más importantes que hay, y se compone de:

- **Name**: Vendrá ya rellenado con el nombre que pusimos cuando renombramos el Form
- **Caption**: Es el nombre que aparecerá en la ventana cuando ejecutemos el programa.
- **Starting Form**: Como es el primer módulo del proyecto, por defecto viene marcado, ésto se usa porque si suponemos que tenemos varios módulos en el proyecto, uno de ellos ha de ser el principal, (normalmente sería la pantalla de inicio), si está marcado y además en gris, es porque ya está asignado a otro form. Con el marcado, lo que le indicamos a PWC, es que cuando ejecute el programa y se encuentre con varios Módulos y Forms, cuál es el que debe de ejecutar por defecto.
- **Enabled**; indica si está habilitado o deshabilitado el Form. (la propiedad *Enabled**,*** habilita o deshabilita el form o cualquier otro elemento o control).
- **Printable**: Indica si cuando hagamos un volcado de pantalla, se imprimirá el Form o no, también veremos más adelante ésta propiedad.
- **MultipleInstance**; indica si podremos abrir el Form más de una vez al mismo tiempo, en este caso, no lo marcamos, o nos dará un error, COBOL85, no soporta múltiples instancias de un mismo form, sólo lo soporta NetCOBOL y no es el caso.
- **Visible**: Indica si el Form va a ser visible o no, veremos esta propiedad más adelante.
- **Window State**; *Normal/Maximize/Minimize*, yo suelo dejarlo en normal y ajusto el tamaño de la ventana manualmente, porque si hacemos una ventana de por ejemplo 800 x 600 y luego la maximizamos a por ejemplo 1280 x 1024, (re-escalado) no queda muy bien, el escalado de objetos, no es el punto fuerte de PWC.
- **Icon**: podemos seleccionar el tipo de icono que vamos a utilizar en el programa, de entre los que vienen por defecto o uno propio.
- **MousePointer**; Exactamente igual que con Icon, pero con el puntero del ratón.
- **Moveable**; Si desmarcamos esta opción, el usuario final no podrá mover la ventana del sitio en que se la coloquemos.
- **Topmost**; siempre al frente, este form estará siempre por encima de cualquier otra ventana que haya abierta y que no pertenezca a este proyecto. (Un navegador web, una hoja de cálculo, …)



Pestaña Coordinates;

En esta pestaña, sólo suelo cambiar un par de datos, *ScaleMode*, que la suelo poner en píxeles, porque es la medida estándar de una pantalla y *StartUpPosition*, donde se le indica al form dónde debe de abrirse, que selecciono siempre *1 – Center Owner*



Scalable, no lo suelo utilizar por lo que he dicho antes, el escalado no es el fuerte de PWC, así que no lo marco, y el resto lo manejo directamente desde el form.

Pestaña Style;

Aquí damos el estilo a la ventana, el borde, los botones, etc. lo mejor es “toquitearlo” todo un poco y ver el resultado. El campo *MenuBarName*, de momento no nos dejará hacer nada, porque aún no hemos creado ninguna barra de menú.

Pestaña StatusBar:

Sólo la utilizaremos, si deseamos que en la parte de abajo de la ventana, nos aparezca una barra de estado, y si es así, podemos indicarle el texto que queremos que aparezca, aunque luego ese texto, podemos cambiarlo por programación cuando queramos. Es parecido a la barra de estado de los navegadores-

Pestaña Resource:

Aquí le indicaremos que icono, flecha de ratón y cadena de imágenes para la barra de menú vamos a utilizar, pero como de momento no hemos cargado ninguna imagen, no la vamos a usar.

Pestañas OLE y Compatibility*:*

Estas dos pestañas, mejor no tocarlas, porque se usan para el control del propio Windows, sobre todo.

Pestaña Colors:

Aquí definiremos los colores de la ventana, (o del control si es el caso), siempre podremos elegir (en el apartado *ColorSet),* entre;

- **System Colors**: Colores del sistema, (los colores de Windows).
- **Standard Colors**: Colores estándar de una paleta de 16 colores
- **Custom Color**: Definiremos, la mezcla RGB o color, brillo y saturación a nuestro gusto.

Y en *Properties:*

- **Backcolor**: Es el color de fondo, tanto para la ventana, como sería en este caso, como para un texto, una tabla, etc.
- **Forecolor**; Es el color del texto, para cualquier elemento, igual que el anterior.
- **HighLightColor**: Es el color de énfasis para cualquier elemento.

Pestaña Fonts:

Aquí definiremos, el tipo y forma de letra que deseamos utilizar, hay que tener en cuenta que, como en éste momento, estamos modificando el form y no un control, (un botón, una tabla, etc.), esta propiedad es heredable, es decir todos los elementos que insertemos en el form, usarán por defecto lo que aquí definamos, aunque luego se puede cambiar para cada uno por separado.

IMÁGENES ESTÁTICAS y/o DINÁMICAS.

Todo programa que se precie, muestra imágenes, ya sea en los botones, como en los menús, fondos, iconos, etc. y hay 2 maneras de tener las imágenes en un programa, una es de manera dinámica o en tiempo de ejecución y la otra es estática o “incrustada” en el programa, ambas tienen sus ventajas y sus inconvenientes.

Dinámica:

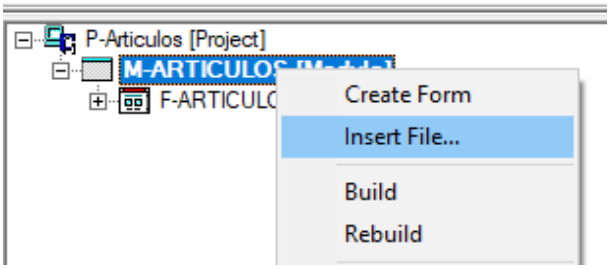
O en tiempo de ejecución, mientras el programa está en marcha, somos nosotros, o el usuario del programa, los que indicamos el nombre y la ruta del fichero que contiene la imagen. Normalmente se utiliza para las imágenes de los artículos, de un cliente, etc. son imágenes que pueden cambiar con facilidad. Como ventaja, tiene que podemos cambiar la imagen cuando queramos, como inconveniente, que nos la pueden cambiar cuando quieran, como por ejemplo la imagen de un botón, el logo de publicidad de nuestra empresa, etc., además de que tenemos que controlar en el programa, que es una imagen en un formato correcto, que existe y algunas cosas más.

Estática:

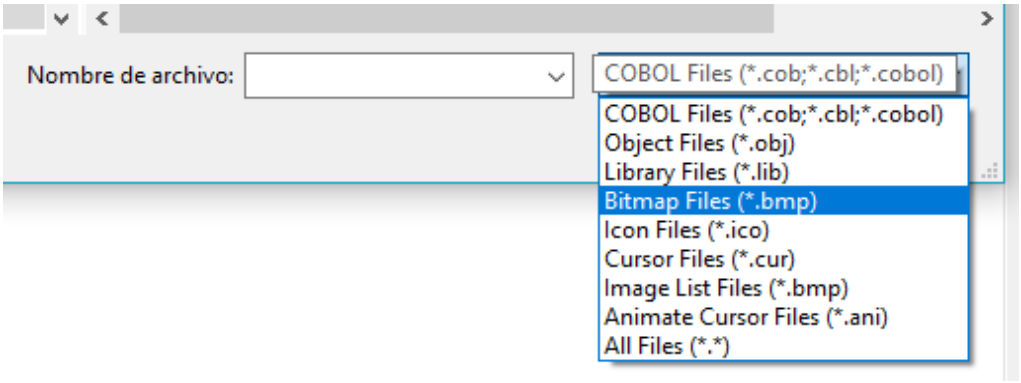
O incrustada, esto se hace añadiendo la imagen al programa y al compilar éste, la imagen queda “incrustada” dentro de él. Normalmente se utiliza para los botones, el logo de publicidad de nuestra empresa, iconos y cosas por el estilo, es decir, imágenes que normalmente, siempre van a ser las mismas. Como ventaja, tenemos que no nos pueden cambiar la imagen, que somos nosotros los que controlamos cualquier aspecto sobre ésta, como desventaja,al incrustar la imagen ésta hace que el tamaño del programa aumente, tanto como “pese” cada una de las imágenes, por lo que si tenemos muchas imágenes, nuestro programa, aumentará considerablemente de tamaño.

En nuestro caso, para los botones, fondos, etc. vamos a utilizar imágenes estáticas, para ello, dentro de la carpeta donde tenemos los fuentes del programa, vamos a crear una carpeta que se llame *Img*, donde vamos a guardar las imágenes que vamos a utilizar. Por desgracia, PWC únicamente admite como imágenes incrustadas las de formato .BMP, .ICO, (Iconos) y .CUR, (Cursores), no admite el estándar más utilizado como .JPG,, pero incluso el Paint, puede convertir cualquier imagen .JPG en .BMP., en el caso de imágenes dinámicas, el formato soportado es algo superior, (aunque no mucho), admite .JPG, .BMP, .PCD, .WMF, .EMF y alguno más que se me olvidará.

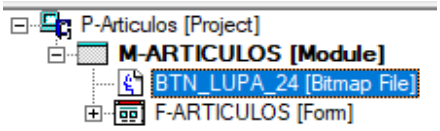
Para incrustar una imagen, hacemos click con el botón derecho en la rama del árbol de nuestro proyecto donde pone [Module] y seleccionamos Insert File ..



y en la parte inferior derecha de la ventana que se nos abre, seleccionamos Bitmap Files (*.bmp)



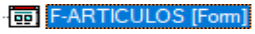
Buscamos el archivo que deseamos incrustar, (en mi caso uno que se llama Lupa_24.bmp, y una vez seleccionado, volveremos al árbol del proyecto y queda a la espera de que cambiemos el nombre por defecto, (en este caso Bitmap1), por un nuevo nombre, teniendo en cuenta que, el nombre que pongamos es el que tendremos que utilizar más adelante en la configuración del botón, como siempre, recomiendo poner un nombre que nos “suene a algo”, yo en mi caso voy a renombrarlo como *BTN_LUPA_24*. El árbol de proyecto os debe quedar así más o menos.



Ésto tendremos que hacerlo con cada una de las imágenes e iconos que vayamos a incrustar en el programa, (Los botones de Aceptar, cancelar, borrar, imprimir, salir, listar, exportar, imágenes de fondo, iconos, etc.), así que ánimo y hacerlo ahora.

Diseño del Form:

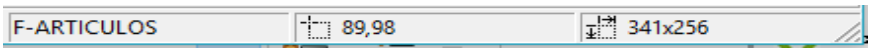
Teniendo en cuenta, que es lo que va a ver el usuario final, el diseño del Form ha de ser lo más claro y limpio posible, así que vamos a ello.

Si hacemos doble-click sobre la rama del form 

, se nos abrirá este y entraremos en lo que comúnmente se llama "*modo diseño*", al momento, se nos abrirá un form "en blanco", además, a la izquierda tendremos un menú vertical, en el que tendremos todos los elementos que podemos insertar en el form, (controles), desde un simple texto hasta un control para escuchar música.

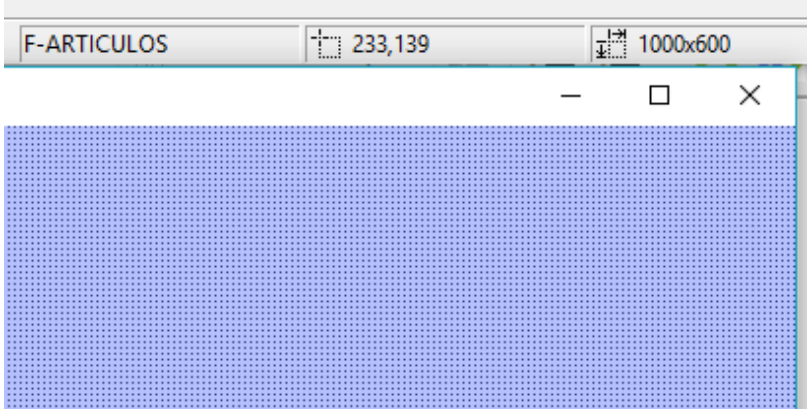
Barra de controles

Lo primero que tenemos que hacer es cambiar el tamaño del form, puesto que el que aparece por defecto, no es lo bastante grande para incluir todos los campos que vamos a usar. El tamaño se cambia como el de cualquier ventana en Windows, ponemos el ratón en cualquier esquina, pulsamos y, sin soltar arrastramos hasta tener el tamaño deseado. Tanto para la ventana del form, como para cualquier otro control, en la parte superior izquierda, tenemos 3 campos de información



El primero, nos indica qué elemento estamos configurando, **el segundo** nos indica las coordenadas de posición de ese elemento, (el primer número es la distancia desde la izquierda y el segundo número es la distancia desde arriba, tomando como punto de origen, 0,0 que es la esquina superior izquierda del form si es un elemento, o de nuestro monitor si es el propio form lo que estamos modificando), y **el tercero**, nos indica el tamaño del elemento, (el primer número es la altura y el segundo número es el ancho).

En mi caso, voy a crear un form con un tamaño de 1000x600 píxeles, al que voy darle un fondo, (BackColor), de color azul claro, el tamaño está puesto a ojo y, una vez terminado el form, lo ajustaremos a su tamaño final.

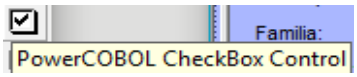


Antes de empezar a incrustar controles, vamos a repasar los de uso más común y sus propiedades. Para ver las propiedades de cualquier control, una vez que lo hemos añadido al form, hacemos doble-click sobre dicho control o pulsamos con el botón derecho y seleccionamos *Propiedades*, Hay muchas propiedades, que son comunes en todos los controles, por lo que no voy a explicarlas en cada uno de ellos. Pestañas como la de *Colors* y *Fonts*, ya las he explicado anteriormente y, salvo sutiles cambios, es igual para todos.

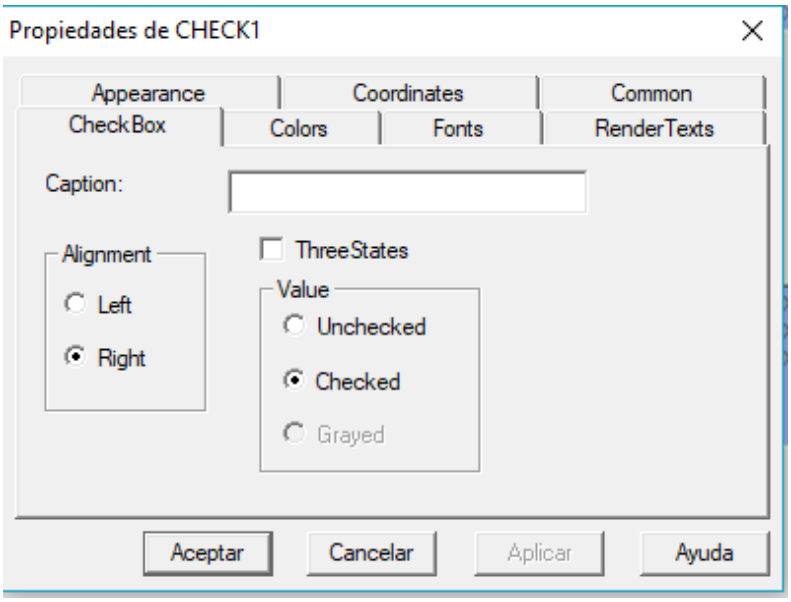
Todos los controles que trae por defecto PWC, los tenemos en la barra vertical de la izquierda que he comentado antes, podemos añadir o quitar controles de la barra. PWC, acepta perfectamente controles de terceros, aunque eso no garantiza que funcionen correctamente, o incluso que funcionen y, por desgracia, la única manera de saberlo es probándolo. Pero tener en cuenta algo muy importante, si insertáis un control en PWC, y utilizais ese control en el Form, si este no funciona o funciona, pero no como esperabais, y, como sería normal, lo borráis del equipo, pero no eliminái s el componente del form, cuando vayáis a abrir de nuevo el proyecto, os dará error de que le falta el control, lo cual no es muy importante, el peligro está en que no os dejará volver a salvar el proyecto hasta que no hayáis solucionado el problema, y ésto si que puede ser un problema y muy gordo, así que, si usáis controles de terceros, procurar tenerlos siempre a mano, por lo que pueda pasar y ante todo, documentar el proyecto muy bien.

LOS CONTROLES de PWC.

PowerCOBOL CheckBox Control:



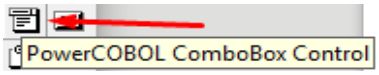
Inserta un checkbox en nuestro formulario.



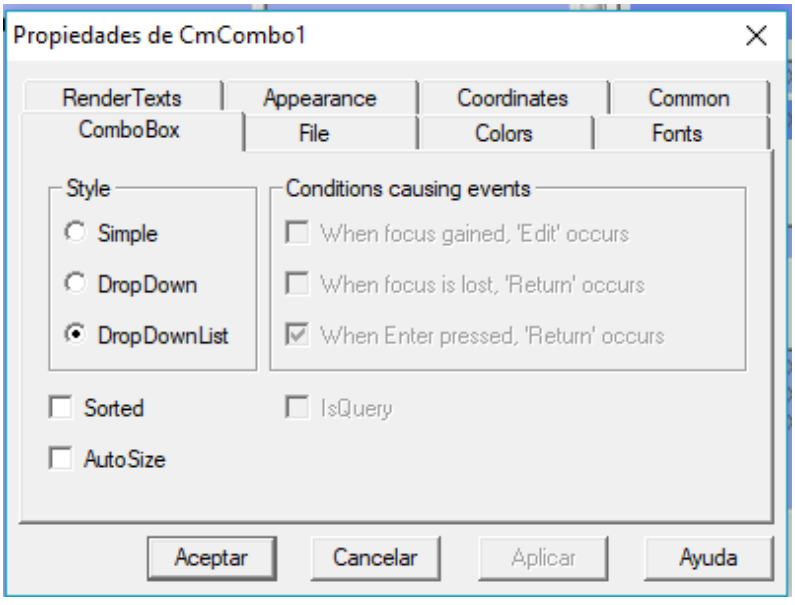
Pestaña CheckBox:

- **Caption:** Es el texto que queremos que nos muestre.
- **Alignment**: *Left*/*Right*, le indicamos si queremos el texto a la izquierda o a la derecha del checkbox.
- **ThreeStates**: Indicamos si queremos que el checkbox tenga 3 estados en vez de 2, (marcado, desmarcado o gris)
- **Value**: Indica cuál es el estado por defecto del checkbox.

PowerCobol ComboBox Control:



Inserta un ComboBox, (menú desplegable), en nuestro formulario.



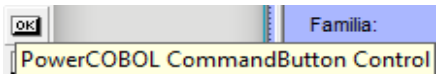
Pestaña ComboBox:

- **Simple**: Es el más sencillo de todos, siempre muestra el combo.
- **DropDown**: Muestra una flecha hacia abajo en la parte derecha del combo, además, podemos seleccionar o filtrar cualquier elemento, si empezamos a escribirlo.
- **DropDownList**: Igual que el anterior, pero sin el filtro.
- **Sorted**: Nos ordena alfabéticamente la lista.
- **AutoSize**: Ajusta el tamaño del desplegable a la cantidad de elementos. (Más o menos).
- **Conditions causing events**:
  - **When focus gained …**: Cuando el control recibe el foco, se genera un evento *Edit*.
  - **When foccus is lost …**: Cuando el control pierde el foco, se genera un evento *Return*, (Tecla Enter)
  - **When Enter pressed …**: Cuando pulsamos la tecla *Enter*, se genera un evento *Return*.
- **IsQuery**: Filtra la lista por los caracteres que escribamos en el cuadro de texto.

Pestaña File:

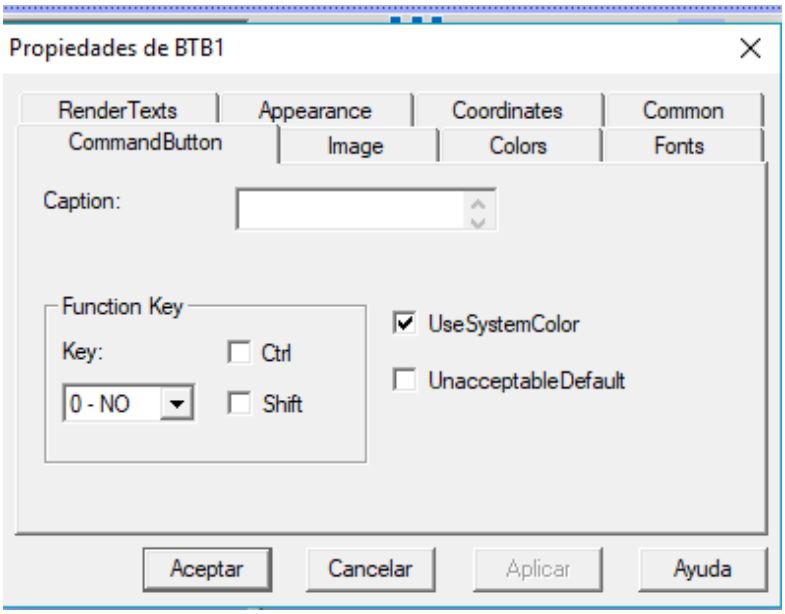
- **File Type**: Le indicamos si la lista, la tiene que cargar desde un fichero, en cuyo caso el fichero puede ser:
  - **Standard Text File**: Fichero de texto estándar. (Texto plano)
  - **CSV Format**: Fichero en formato CSV.
  - **Fixed Length Format**: Fichero de texto plano con longitud fija.
  - **Record Form**: Si es texto de longitud fija, le indicamos la longitud de los campos, los campos pueden estar separados por espacios o comas, (,), por ejemplo, para especificar 2 elementos de 11 y 8 caracteres, sería `MOVE "11,8" TO "RecordSet" OF …`
- ***CurField**: Le indicamos el número de elementos que debe de cargar por fila, cuando es un fichero CSV o de longitud fija.*
- **FileName**: Nombre del fichero que contiene los datos.
- **AutoLoadFile**: Le indicamos si debe de cargar el fichero automáticamente al cargar el form o no.

PowerCOBOL CommandButton Control



,

Inserta un pushButton (Botón normal) en el form.



Pestaña CommandButton:

(Todas las opciones de ésta pestaña, se pueden cambiar en tiempo de ejecución).

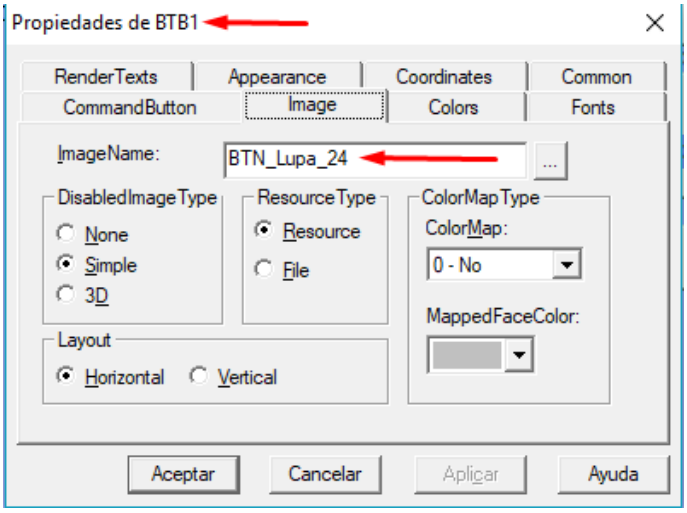
- **Caption**: Es el texto que aparecerá en el botón, (no es obligatorio), además, el texto puede llevar el mnemónico "**&**", como tecla de acceso rápido.
- **Function Key**: Podemos asignarle una o una combinación de teclas al botón como acceso rápido.

- **UseSystemColor**: Le indicamos que utilice los colores por defecto de Windows, si lo marcamos, se ignorará cualquier cambio que hagamos en la pestaña *Colors*. Si no lo marcamos, el color del botón, será el que seleccionemos en la opción *BackColor* de la pestaña *Colors* y el texto con el que pongamos en la opción *ForeColor*, el color de resaltado y de sombreado, será definido por el sistema.
- **UnacceptableDefault**: Si marcamos ésta opción, será el botón por defecto de nuestra aplicación.

Pestaña Image:

(Todas las opciones de ésta pestaña, se pueden cambiar en tiempo de ejecución).

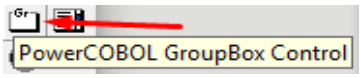
- **ImageName**: Nombre de la imagen, si es incrustada el nombre que le hayamos puesto al incrustarla en el proyecto y si es dinámica, con el botón que hay a la derecha, (…), la buscamos y la seleccionamos.
- **DisabledImageType**: Cómo se verá la imagen si el botón está deshabilitado.
- **Resource**: Si la imagen es incrustada, seleccionamos ésta opción.
- **File**: Si la imagen es dinámica seleccionamos ésta opción.
- **ColorMap**: Con esto cambiamos el color del botón, es bastante difícil de explicar las opciones y exclusiones que tiene de forma sencilla, por lo que, la mejor manera es probándolo.
- **MappedFaceColor**: Se usa junto con la opción anterior.
- **Layout**: *Horizontal*/*Vertical*: Si el botón tiene un texto y una imagen, Indica donde queremos colocar el texto, horizontal es a la derecha de la imagen y vertical es debajo.



En nuestro caso, la pestaña *Image* queda así:

Si os fijáis, también he cambiado el nombre del botón por *BTB1*, en la pestaña *Common*.

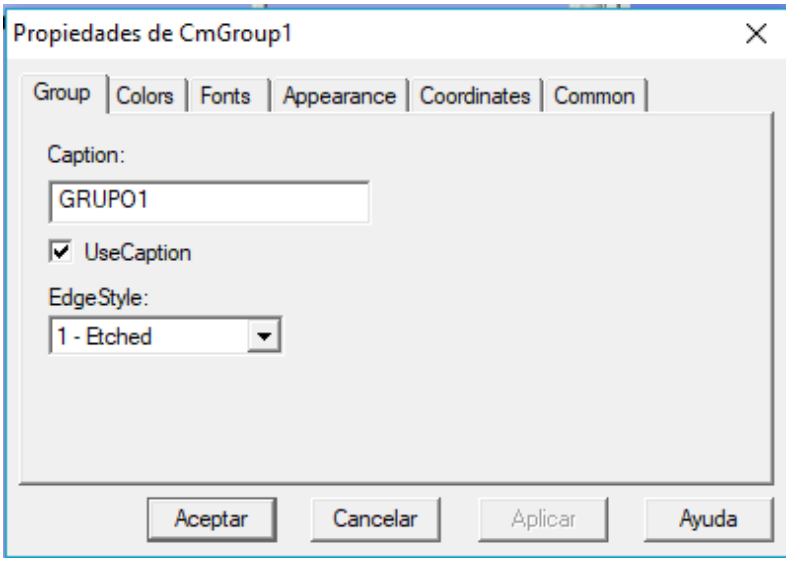
PowerCobol GroupBox Control



Sirve para agrupar varios controles juntos, por ejemplo varios *OptionButton*, botones, etc., si inhabilitamos el *GroupBox*, inhabilitamos todos los elementos que lo componen, es una ventaja, pero también tiene como inconveniente que, para hacer referencia a cualquier elemento incluido en el *GroupBox*, hay que hacer referencia a ambos, (el elemento y el grupo), por ejemplo

```
MOVE 1 TO "Enabled" OF BOTON1 OF GRUPO1
```

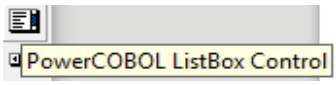
y si son muchos elementos, es pesado, por otro lado si vamos a utilizar varios grupos distintos de *OptionButton Control*, debemos a la fuerza crearlos en distintos Grupos. Como información muy importante, es que si vamos a usarlo, primero hay que crear el *GroupBox* y después insertarle los controles que queramos dentro de él, porque si lo hacemos al revés, por ejemplo, si vamos a crear 2 botones dentro de un *GroupBox* y, creamos primero los botones, escribimos el código fuente para esos botones, creamos el *GroupBox*, y queremos meter los botones dentro de ese grupo, no nos deja, (si intentamos "arrastrarlos" dentro del *GroupBox, nos aparecerá un símbolo de prohibido),* a no ser que "*Cortemos*" y "*Pegemos*", (Ctrl.+ X y Ctrl. + V) los botones dentro del grupo, pero, ésto hará que todo el código fuente que hayamos escrito para los botones, se pierda, (y así nos lo indicará PWC), lo cual, puede ser un problema muy gordo, si el código fuente es muy largo o especialmente complicado. Esto es así para los GroupBox, FrameBox y TabControl, (muy importante en éste ultimo control, porque lo vamos a utilizar en nuestro programa, ya que será la base sobre la que estén colocados la mayoría de los controles, como campos informativos, campos para introducir texto, botones de búsqueda, etc.)



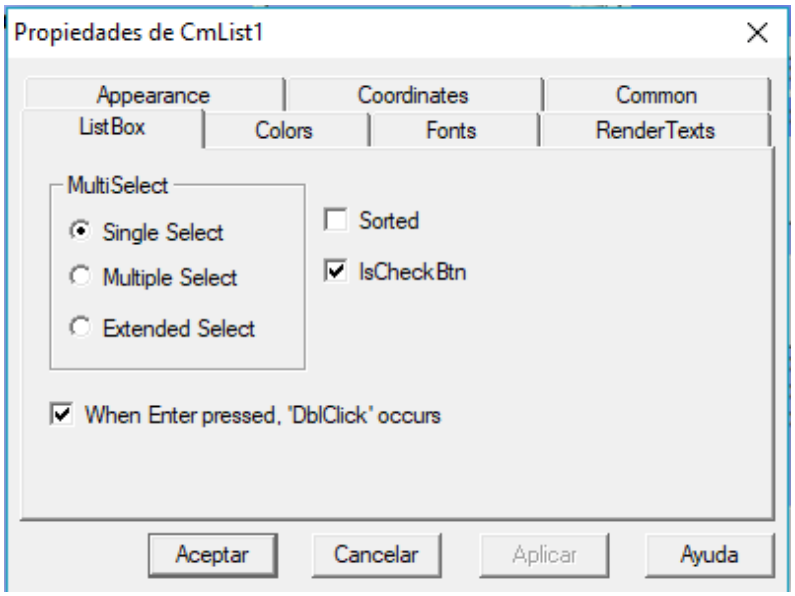
Pestaña Group:

- **Caption**: Es el texto que aparecerá en la parte superior izquierda del control.
- **UseCaption**: Si lo marcamos, aparecerá el texto incluido en *Caption*.
- **EdgeStyle**: Es el estilo que queremos darle al borde del grupo.

PowerCobol ListBox Control



Inserta una lista dentro de una “caja” en nuestro formulario. (Tipo a las que se usan para pasar “de la izquierda” a “la derecha”).



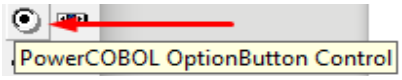
Pestaña ListBox:

- **MultiSelect**: Indicamos que tipo de selección vamos a utilizar.
  - **Single Select**: Sólo podremos seleccionar un elemento.
  - **Multiple Select**: Permite seleccionar más de un elemento. (Con la tecla *Ctrl*.)
  - **Extended Select**: Igual que el anterior, pero permite seleccionar también entre rangos. (Con las teclas *Alt+Ctrl*)
- **Sorted**: Indica que queremos ordenar la lista alfabéticamente, con la salvedad de que si insertamos un elemento mediante el método “*InsertString*”, (`INVOKE CmList1 "InsertString" USING cadena`

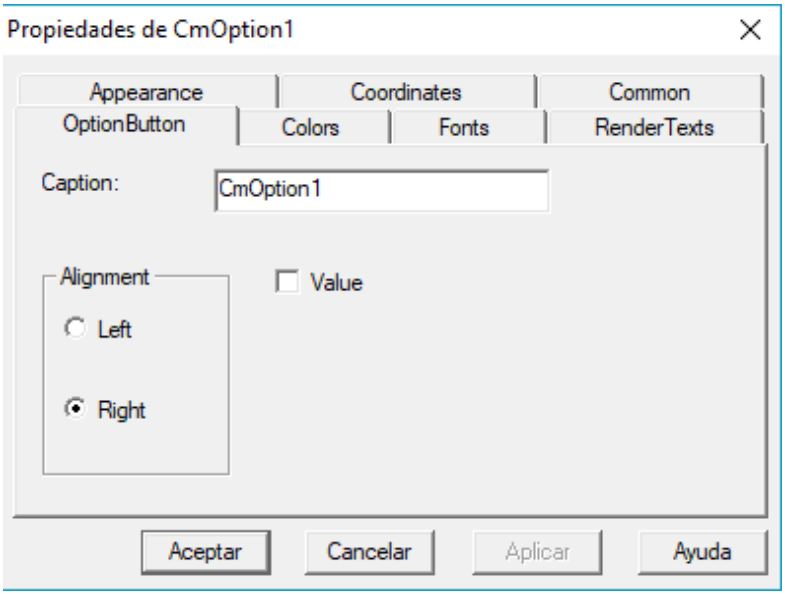
índice.), éste no se ordenará automáticamente, para ello hay que usar el método "*AddString*", (**INVOKE CmList1 "AddString" USING cadena.**).

- **IsCeckBtn**: Muestra un cuadrado *CheckBox*, al lado de cada elemento de la lista.
- **When Enter pressed …**: Cuando pulsamos la tecla *Enter* sobre un elemento, se activa el evento *Doble-click-*

PowerCobol OptionButton Control.



Inserta un "botón" de selección única en nuestro form. (Lo que comúnmente se conoce como un *RadioButton)*.

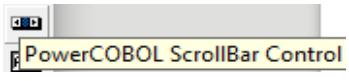


Pestaña Option Button:

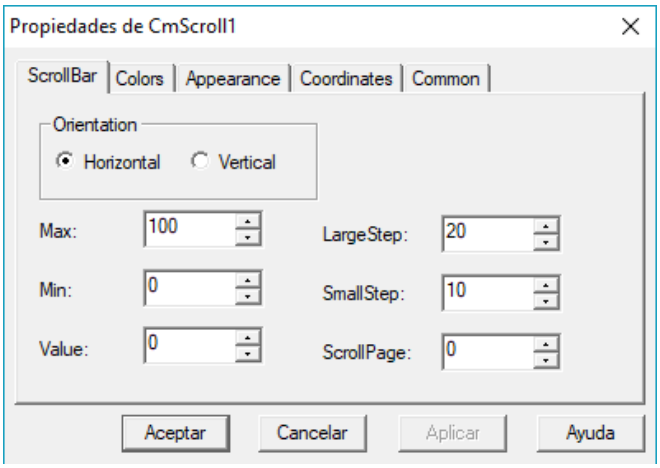
- **Caption:** Es el texto que queremos que nos muestre.
- **Alignment**: *Left*/*Right*, le indicamos si queremos el texto a la izquierda o a la derecha del OptionButton.
- **Value**: Indica que en caso de haber varios, éste sería el marcado por defecto.

Al contrario que el *CheckBox*, aquí sólo puede haber uno marcado a la vez, en todo el form, a no ser que los pongamos dentro de un *GroupBox Control*, que en ese caso, pueden haber tantos marcados como *GroupBox* tengamos.

PowerCobol ScrollBar Control



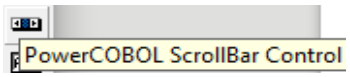
Inserta una barra de desplazamiento en el form.



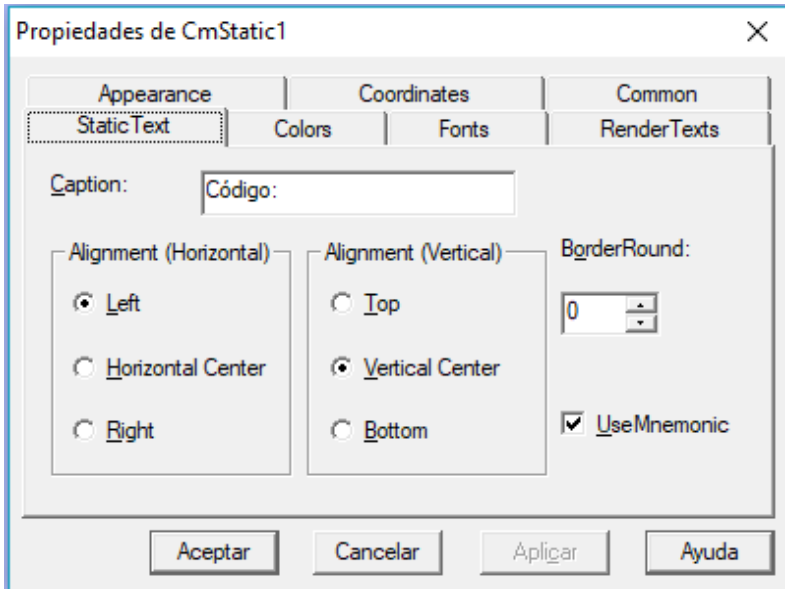
Pestaña ScrollBar:

- **Orientation**: Coloca la barra en horizontal o en vertical.
- **Max**: Corresponde con el valor máximo que queremos asignar a la barra.
- **Min**: Corresponde al valor mínimo.
- **Value**: Indicamos el valor inicial.
- **LargeStep**: Es el valor de desplazamiento, cuando hacemos click dentro de la barra entre las dos flechas.
- **SmallStep**: Es el valor de desplazamiento, cuando hacemos click en una de las dos flechas.
- **ScrollPage**: Define el tamaño de la barra de scroll , El tamaño, viene calculado por la fórmula: ***(Max - Min) / ScrollPage***.

PowerCobol StaticText Control



Inserta un control de texto estático en nuestro form. Es uno de los controles más usado en un form. Son textos que, supuestamente no van a cambiar y además no son editables directamente, (por el usuario final), son tipo a, por ejemplo, "*Nombre:*", "*Razón Social*:", "*Código*", "Descripción", etc.



Pestaña StaticText:

- **Caption**: Es el texto que va a aparecer en el form, puede contener hasta 8192 caracteres.
- **Alignment Horizontal/Vertical**: Corresponde a la alineación que queremos darle al texto.
- **BorderRound**: En el caso de que le asignemos un borde al elemento, podemos redondear las esquinas.
- **UseMnemonic**: Se usa para indicar si, en el caso de que utilicemos el carácter "**&**", si éste va a ser el de acceso rápido o si forma parte del texto. El carácter &, se usa en programación de formularios para indicar el acceso rápido a un campo, (es el que sale subrayado cuando en un programa pulsamos la tecla *Alt*).

Pestaña RenderTexts:

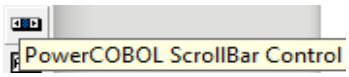
Esta la explicaremos con más detalle, cuando configuremos el control *PowerCOBOL TextBox Control*.

Pestaña Appearence:

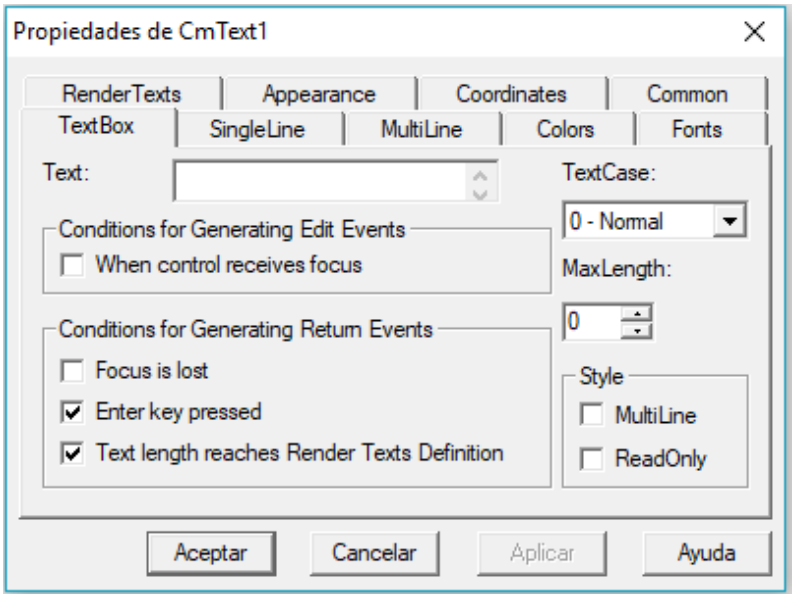
- **BackStyle**: Pude ser Transparent, el control muestra el color de fondo, en éste caso el del ControlTab, u Opaque, el control muestra el color que definamos en la pestaña Colors.
- **BorderStyle**: *Off* o *Solid*, no muestra ningún borde o si lo muestra.

- **Appearance**: *Flat/3D*, cambia la apariencia entre plana o 3D
- **Enabled**; Si está marcado, lo muestra en color normal, si está desmarcado, lo muestra en un tono gris que indica que está deshabilitado.

PowerCobol TextBox Control



Inserta un control de texto editable en el form. Es uno de los controles más importante y más usado, es en estos controles, donde el usuario final, introduce los datos al programa.



Pestaña TextBox:

- **Text**: Es el texto que queramos que tenga por defecto el campo, vendría a ser el equivalente a la cláusula **VALUE** en COBOL. Puede tener una longitud máxima de 8192 caracteres.
- **When control receives …**: Determina si se ha de producir el evento Edit, cuando el campo recibe el foco, (o cuando se rellena por completo), esto hace que si el evento Edit, tiene algo programado, se ejecute al menos una vez, si no introducimos nada en él.
- **Conditions for Generating Return …**: Son en que condiciones especiales, se va a generar el evento Return
  - **Focus is lost**: Cuando ese campo pierda el foco, (con esto el evento Return, se ejecutará al menos una vez).
  - **Enter key pressed**: Cuando pulsemos la tecla Enter/Intro, es lo normal, que al pulsar la tecla Enter, cambiemos al siguiente campo, por ejemplo.
  - **Text length reaches …**: Cuando el texto llegue al tamaño máximo del campo, equivaldría a la cláusula **AUTO** de COBOL.
- **TextCase**: *Normal*/*UpperCase*/*LowerCase*, el texto que introduzcamos sería normal, convertido a mayúsculas o convertido a minúsculas. Equivaldría a las clausulas **UPPER** y **LOWER** de COBOL.
- **MaxLength**: Si el valor es 0, (cero), el tamaño del campo, se ajustará al que definamos en la cláusula PIC del campo, de lo contrario, se ajustará al valor que pongamos aquí, equivaldría a la cláusula **SIZE** en COBOL..
- **MultiLine**: Si el campo va a tener varias lineas o no, en estas condiciones, cuando pulsemos la tecla Enter/Intro, no se generará el evento Return
- **ReadOnly**: Hace que el campo sea de sólo lectura.

Pestaña SingleLine:

- **Select Text**: Cuando el campo recibe el foco, queda todo el texto autoseleccionado. (Marcado en azul).
- **Password**: Enmascara el texto, para que no se vea lo que escribimos, equivaldría a la cláusula **PROMPT "*"** de COBOL
- **Password Char**; Carácter con el que queremos que se enmascare el texto.
- **Editable Label**: Determina si el campo entra en "modo edición" en cuanto hagamos click en él o no. En mi caso, y en el de muchos ejemplos que he visto, casi nadie utiliza esta opción, porque suele ser bastante problemática a la hora de controlarlo y de poder controlar la tecla Enter.

- **AutoEdit**; Indica si el campo va entrar en modo editable automáticamente o no.
- **DecisionMaxString**: Determina si la introducción de texto ha sido completada cuando llegamos al tamaño máximo del campo,
- **Alignment**: *Left*/*Center*/*Right*, Izquierda, centro, derecha, alineación del texto en el campo, esto sólo podremos configurarlo si **Editable Label**, está marcado.

Pestaña Multiline:

- **Horizontal/Vertical ScrollBar**: Si en la pestaña anterior hemos seleccionado *MultiLine*, podemos hacer que nos aparezcan unas barras de desplazamiento tanto en horizontal como en vertical.
- **AutoScroll**: Cuando escribimos un texto y llega al final, cambiará de linea automáticamente y hará un scroll vertical automático.

Pestaña RenderTexts:

(Ninguna de las opciones de ésta pestaña, se pueden cambiar en tiempo de ejecución).

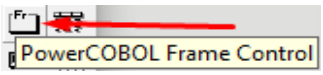
- **Properties**: De momento no podemos modificarlo, sólo nos dejará cuando sea un *tableControl* o *listView*.
- **RenderStyle**: *Standard*/*COBOL Picture*/*Date*:
  - **Standard**: El texto quedará tal y como lo escribamos, sin formatear.
  - **COBOL PICTURE**: En el campo *PictureString*, introduciremos el texto como queremos que quede formateado, exactamente igual que lo haríamos en COBOL y además con las mismas normas. Hay que tener en cuenta que, si en el campo *Text* de la pestaña *TextBox*, tenemos introducido un texto alfanumérico, y aquí definimos un texto numérico, (tipo Z.ZZZ.ZZ9,99 por ejemplo), nos dará un error y no nos dejará continuar hasta que lo hayamos solucionado
  - **Date**: Definiremos un campo para los formatos de fechas, tiene la ventaja de que no tendremos que comprobar si la fecha es correcta o no, lo realizará el PWC por nosotros. Además, como veremos a continuación, podremos definir ampliamente el formato.
- **Picture Style**:
  - **CurrencySign**; Definimos el símbolo monetario que queremos usar, (€,$,¥, etc).
  - **DecimalPointIsComma**: Exactamente igual que en COBOL, aunque por experiencia propia, tanto en esta versión como en la versión 7.n de PWC, da lo mismo lo que pongamos aquí, como no esté definida en la sección **SPECIAL-NAMES** de nuestro programa, no funcionará.
  - **BlankSupress**; Igual que en COBOL, si el campo tiene el valor 0, (cero), aparecerá en blanco.

Las siguientes opciones, sólo estarán disponibles si hemos seleccionado *Date* en *PictureString*.

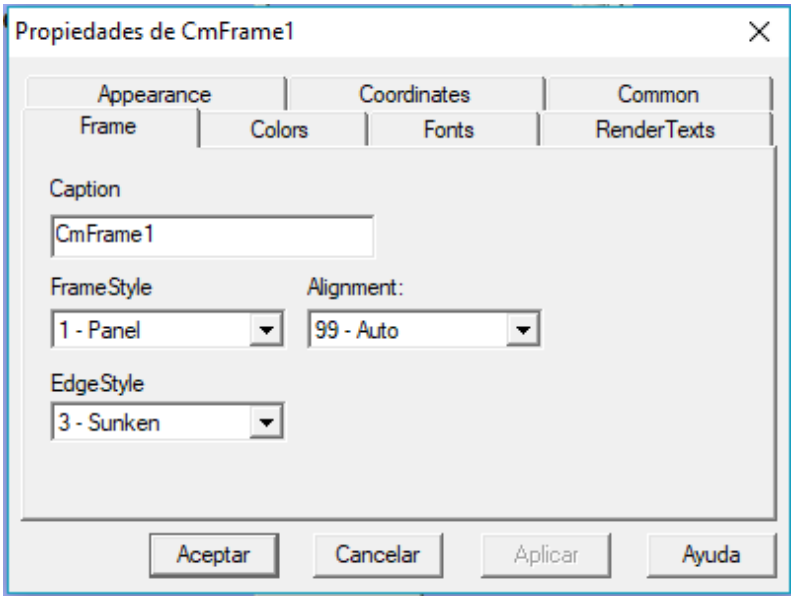
- **InputFormat**: Define cómo vamos a introducir la fecha. Podemos elegir cualquiera de las que vienen por defecto.
- **DateStyle**: Podemos elegir una de las predefinidas o elegir la última (*Custom*), si elegimos la última, en el campo *DateFormat*, tendremos que definir el formato de fecha que queremos, ateniéndonos a las siguientes convenciones:
  - Podemos usar como separador el carácter que queramos, da lo mismo que sea una barra, "/", un punto, ".", que una string "abcde", lo que pongamos colará.
  - Podemos incluir los caracteres que queramos donde queramos, por ejemplo, *Firmado el dd de MMM de yyyy*, nos devolverá: Firmado *el nn de Mes de nnnn*.
  - Las convenciones para los campos día, mes y año son:
    - **dd**; para el día, tal y como está aquí escrito, de lo contrario, nos devolverá una string con el texto.
    - **MM**; para el mes en número, tal y como está escrito, igual que el anterior.
    - **MMM** ó **Mon**: para las tres primeras letras del mes, (en inglés).
    - **MMMM** ó **Month**: para el nombre del mes completo, (en inglés).
    - **yyyy**; para el año en 4 dígitos, (no se puede poner el año en 2 dígitos).
- **UseDefaultDate**: Por defecto introduce automáticamente la fecha del sistema.

A los campos estáticos, no suelo cambiarles el nombre, a no ser que sea un campo donde muestro alguna información, y dejo el que pone PWC por defecto, (CmStatic*nnn*), pero como dije más arriba, a los campos de texto editables, sí que les cambio el nombre, empezando por *CAMPO1* el primero y *CAMPOnnn* el último, cambiando la numeración a la siguiente centena, si cambio de pestaña, pero lo dicho allá cada uno. Así que en éste caso, si que voy a ir a la pestaña *Common*, y voy a cambiar en el apartado *Name*: el **CmText***n* que hay por defecto por **Campo***n*, que uso yo. Si no hemos escrito aún ninguna linea de programación, no pasa nada, pero en caso contrario, nos aparecerá una ventana, preguntándonos si queremos cambiar el texto **CmText***n* por el texto **Campo***n* en nuestro form.

PowerCobol Frame Control



Inserta un frame o “marco” en nuestro form. Se utiliza sobre todo para separar los campos por secciones o grupos.



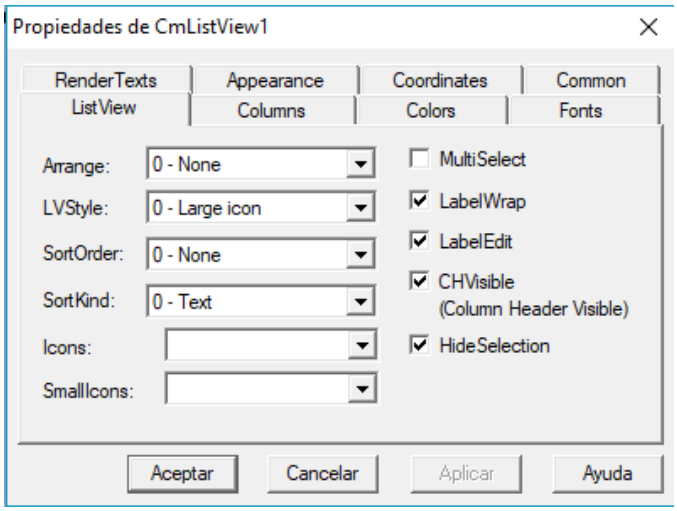
Pestaña frame:

- **Caption**: Es el texto que aparecerá en nuestro control. (Hasta 8192 caracteres).
- **FrameStyle**: Estilo del marco:
  - **GroupBox**: Crea el marco con el texto sobre las lineas del marco.
  - **Panel**: Crea el marco con el texto fuera de las lineas del marco.
- **EdegeStyle**: Definimos la forma del marco que queremos. (Por defecto auto es en la parte superior izquierda).

Al igual que el control *GroupBox*, los elementos que pongamos dentro de éste control, pertenecen a él y son heredados, por lo que no podremos “sacarlos” de ahí, a no ser que “cortemos” y “peguemos” el elemento, pero en éste caso, si el elemento, tiene algún código ya programado, lo perderemos. Además las propiedades de los elementos insertados, heredan las propiedades del frame. La mayor diferencia con el *GroupBox*, es que a la hora de escribir el código fuente de alguno de los elementos insertados en el frame, no hay que referenciarlos con el frame, por ejemplo, si tenemos un botón en el frame, el código sería, por ejemplo `MOVE 1 TO "Enabled" OF BOTON1` y no `MOVE 1 TO "Enabled" OF BOTON1 OF FRAME1` como haríamos en un *GroupBox*.

PowerCobol ListView Control

Inserta un control “listview” en nuestro form, (lo que sería como un tipo de “hoja de cálculo”), de los 2 controles que hay de éste tipo, (éste y *TableControl*), es el más utilizado, sobre todo por ciertas limitaciones en el otro control.



Pestaña ListView:

- **Arrange**: Indicamos la posición de los iconos, izquierda o derecha.
- **LVStyle**: Indicamos el formato de *listView* que vamos a usar. (Son un poco difíciles de explicar, por lo que la mejor manera de ver el resultado es probándolo)
  - **Large Icon**: Iconos grandes, con la etiqueta por debajo.
  - **Small Icon**: Iconos pequeños, con la etiqueta a la derecha.
  - **List**: Iconos pequeños, con la etiqueta a la derecha.
  - **Report**: Iconos pequeños, con la etiqueta y otros detalles, a la derecha. (Es el formato más utilizado).
- **SortOrder**: En caso de que queramos ordenación por defecto, indica si va a ser ascendente o descendente.
- **SortKind**: Indica si el campo de ordenación es alfanumérico, (Text) o numérico, (Numeric)
- **Icons**: Nombre de la imagen incrustada que contiene los iconos de tamaño grande. (*Large icon*)
- **SmallIcons**:Nombre de la imagen incrustada que contiene los iconos de tamaño pequeño. (Small *icon*)
- **MultiSelect**: Indicamos que tipo de selección vamos a utilizar.
  - **Single Select**: Sólo podremos seleccionar un elemento.
  - **Multiple Select**: Permite seleccionar más de un elemento. (Con la tecla Ctrl.)-
  - **Extended Selec**t: Igual que el anterior, pero permite seleccionar también entre rangos. (Con las teclas Alt+Ctrl).
- **LabelWrap**: Si el texto no coge en una linea, lo divide en varias lineas o en caso contrario lo corta y le añade tres puntos suspensivos por donde corta, (…).
- **LabelEdit**: Indica si el texto de la etiqueta, podrá ser modificado en tempo de ejecución, si hacemos click en él.
- **CHVisible**: Indica si la cabecera de la columna será visible o no, (por ejemplo, en una hoja de cálculo, sería si veríamos la cabeceras que tienen las letras **A,B,C …**)
- **HideSelection**: Indica si cuando hemos seleccionado una casilla, ésta debe mostrarse como seleccionada, (con el vídeo inverso), si el *ListView*, pierde el foco.

Pestaña Column:

- **Column**: En caso de que haya más de una columna, aquí seleccionamos la columna que queremos modificar.
- **SortColumn**: Indicamos la columna de ordenación por defecto.
- **Text**: Muestra el texto de la columna seleccionada en el anterior campo *Column*
- **Width**: Ancho de la columna seleccionada, (En unidades de medida indicados en la pestaña *Coordinates*)
- **Alignment**: Alineación izquierda o derecha. Si es la columna **1**, (uno), sólo permite la alineación izquierda.

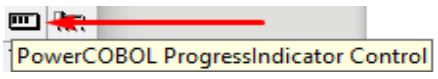
Los botones de ésta pestaña, se utilizan para configurar el *listView*, además, todas las acciones sobe el *listView*, se realizan sobre la columna que tengamos seleccionada en campo *Column.*

- **Insert**: Inserta una nueva columna a la derecha de la que nos encontremos.
- **Remove**: Elimina la columna en la que nos encontramos.
- **Update**: Cualquier modificación que hagamos, ya sea a través de los botones, o a través de los campos *Text* o *Width*, tenemos que confirmarla con éste botón, y además, yo aconsejaría que, una vez pulsado éste botón, pulsemos el botón *Aplicar*/*Apply*, para evitarnos sorpresas desagradables.
- **Clear All**: Elimina todos los campos del *listView*.

Pestaña RenderText:

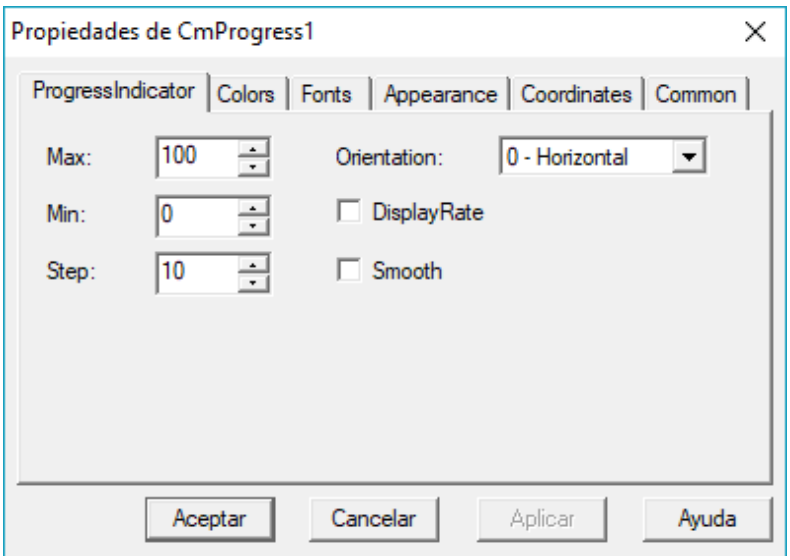
- **Properties**: Aquí tenemos todas las columnas que tiene el *listView*, seleccionamos una y damos el formato que queremos, ateniéndonos a las reglas descritas anteriormente sobre éste apartado.

PowerCobol ProgressIndicator Control



Inserta una "*barra de progreso*" en nuestro form. La barra de progreso que trae PWC, no es precisamente una maravilla que digamos, así que si podéis utilizar un componente de terceros, mejor. Yo, personalmente, utilizo una que se llama XP_Progess_bar_indicator, (***progressbar-xp.ocx*** que, junto con otras librerías y ocxs, adjunto con éste manual), tiene muchas más opciones, es bastante más bonita, no "parpadea" como la de PWC, funciona perfectamente en cualquier versión de Windows, incluido Windows 10 x64 ®, (probado), y, además es gratuita. Tan

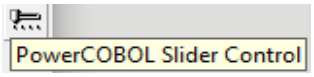
solo, tenemos que copiar el archivo que adjunto *progressbar-xp.ocx*, a la carpeta **C:|Windows\System32** y registrarla con el comando regsvr32.exe y añadir el control a PWC.



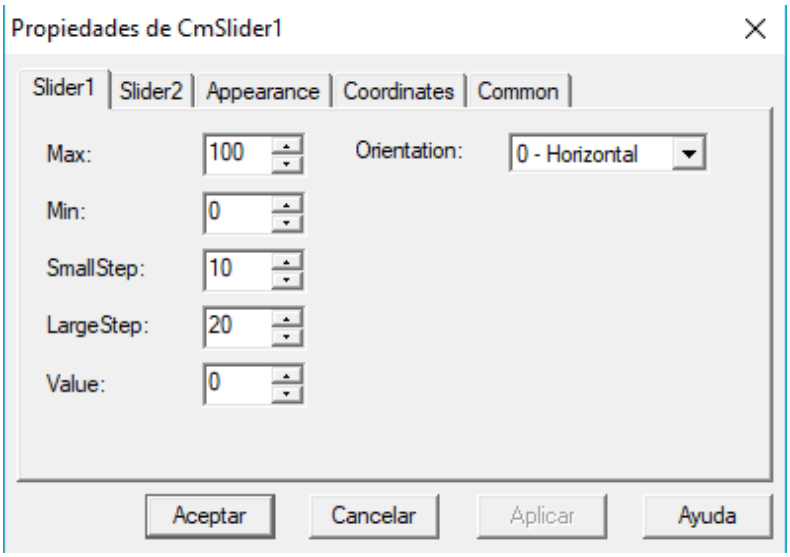
Pestaña ProgressIndicator:

- **Max**: Indica el valor máximo para la barra.
- **Min**: Indica el valor mínimo para la barra.
- **Step**: Indica el valor de cambio para la barra.
- **Orientation**: Indica se queremos una barra horizontal o vertical.
- **DisplayRate**: Indica si además de la barra de progreso, queremos que salga también el porcentaje en números,
- **Smooth**: Indica si la barra va a ser “lisa” o va a estar dividida por una serie de lineas verticales, (las lineas verticales dependen del valor que pongamos en*Step*).

PowerCobol Slider Control



Inserta un control de barra deslizante en nuestro form.



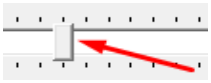
Pestaña Slider1:

- **Orientation**: Coloca la barra en horizontal o en vertical.
- **Max**: Corresponde con el valor máximo que queremos asignar a la barra.

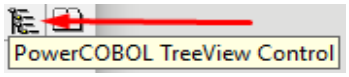
- **Min**: Corresponde al valor mínimo.
- **Value**: Indicamos el valor inicial.
- **LargeStep**: Es el valor de desplazamiento, cuando hacemos click dentro de la barra.
- **SmallStep**: Es el valor de desplazamiento, cuando pulsamos las teclas de las flechas del teclado.

Pestaña Slider2:

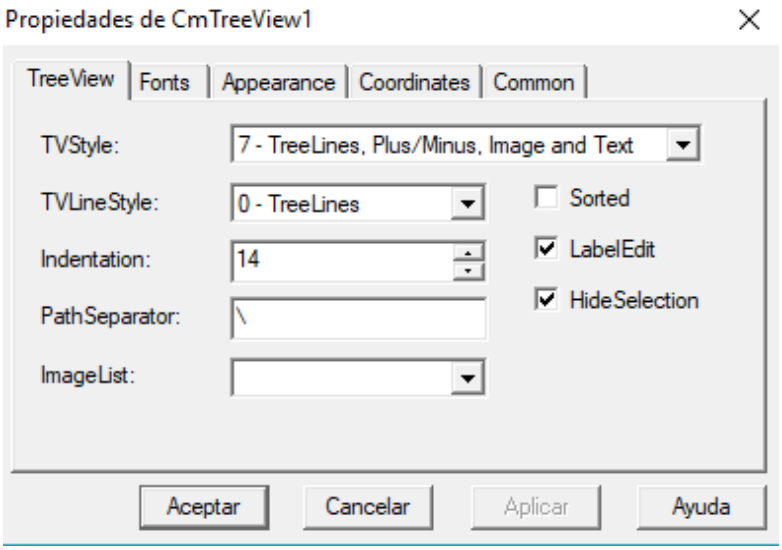
- **AutoTick**: Indica si deben de mostrarse o no las “marcas de graduación” en la barra de desplazamiento.
- **TickFrequency**: Indica el incremento entre las marcas de graduación.
- **TickStyle**: Indica la posición en que han de salir las marcas de graduación:
  - **No Ticks**: Sin marcas de graduación.
  - **Bottom/Right**: En la parte inferior de la barra.
  - **Top/Left**: En la parte superior de la barra.
  - **Both**: En ambos lados.
- **RangeSelect**: Indica si debe marcarse un rango dentro de la barra.
  - **SelectStart:** Inicio del rango de la marca.
  - **SelectEnd**: Fin del rango de la marca.
- **ThumbVisible**: Indica si el “desplazador” va a ser visible o no
- **ValueTips**: Indica si al mover el “desplazador”, será visible el valor de éste o no.


Desplazador.

PowerCobol TreeView Control



Inserta un control de árbol de elementos en nuestro form.

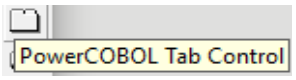


Pestaña TreeView:

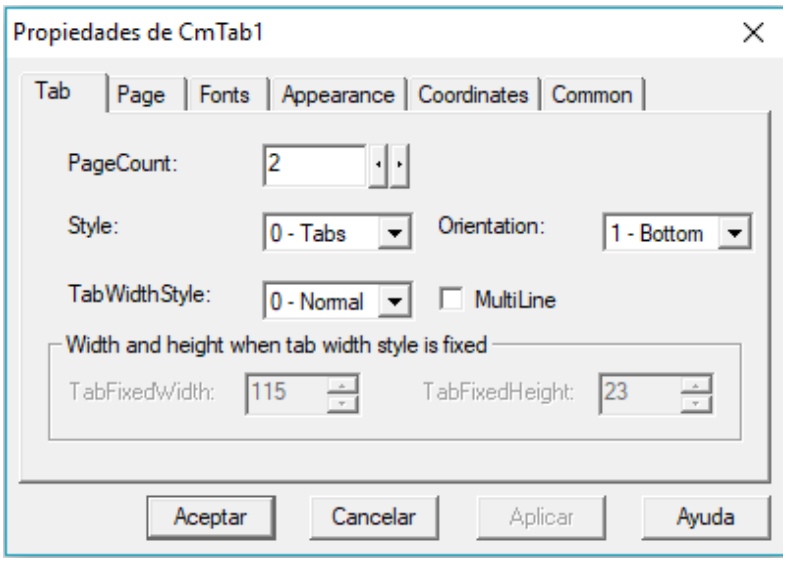
- **TVStyle**: Indicamos el formato de treeView que queremos.
  - **0 - Text Only**: Sólo texto, con todas las ramas abiertas y sin lineas de enlace.
  - **1 - Image and Text**: Texto e imágenes, con todas las ramas abiertas y sin lineas de enlace.
  - **2 - Plus/Minus and Text**: Sólo texto, con las ramas cerradas, los símbolos **+** y **–** para abrirlas y cerrarlas y sin lineas de enlace.
  - **3 - Plus/Minus Image and Text**: Como el anterior incluyendo imágenes.
  - **4 - TreeLines and Text**: Sólo texto y lineas de enlace, con todas las ramas abiertas.
  - **5 - TreeLines Images and Text**: Texto e imágenes y lineas de enlace, con todas las ramas abiertas.
  - **6 - TreeLines, Plus/Minus and Text**: Sólo texto, con todas las ramas abiertas, los símbolos **+** y **–** para abrirlas y cerrarlas y con lineas de enlace.
  - **7 - TreeLines, Plus/Minus, Image and Text**: Como el anterior incluyendo imágenes.

- **TVLineStyle**: Indicamos qué estilo queremos para las líneas de los elementos principales de las ramas (root).
  - **0 - TreeLines**: Los elementos principales, (root), no tienen lineas de enlace entre ellos.
  - **1 - RootLines**:Los elementos principales, (root), sí tienen lineas de enlace entre ellos.
- **Indentation**: Indicamos la cantidad de subramas que va a tener una rama principal, (root).
- **PathSeparator**: Indicamos cuál es el carácter separador entre las ramas y subramas.
- **ImageList:** Nombre de la imagen incrustada que contiene los iconos de las ramas principales, (root).
- **Sorted**: Indica si queremos que las ramas y subramas estén ordenadas alfabéticamente.
- **LabelEdit**: Indica si podemos editar directamente el nombre de la rama, con sólo hacer click en ella.
- **HideSelection**: Indica si cuando hemos seleccionado una rama, ésta debe mostrarse como seleccionada, (con el vídeo inverso), si el tree*View*, pierde el foco.

PowerCobol Tab Control.



Inserta un control de formulario “por pestañas” en nuestro form. Hay que tener en cuenta, que los campos o controles que pongamos dentro de cada una de las pestañas, pertenecen única y exclusivamente a esta, quiere decir que no podremos sacar el control de la pestaña, ni moverlo de una a otra, la única manera de hacer ésto sería borrándolo, (o cortándolo), y volviendo a crearlo, (o pegándolo), donde lo quisiéramos situar de nuevo, por lo que perderíamos el orden y el código fuente que le hayamos programado. Así que si tenéis previsto incluir éste control, procurar que sea de los primeros, porque lo más seguro es que sea éste control el que contenga los campos fijos y editables de vuestro programa.

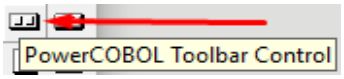


Pestaña Tab:

- **PageCount**: Es el número de pestañas que tendremos. Éste control, no se puede cambiar en tiempo de ejecución, sólo en modo diseño.
- **Style**: *Tabs/Buttons*, indicamos si queremos que el “gestor de pestañas” sea por pestañas o por botones.
- **Tab Width Style**: *Normal/Fixed*, indica si queremos que el ancho de las pestañas, sea normal (el ancho se ajusta al texto) o fjjo, (una anchura predeterminada e igual para todas).
- **Multiline**: Si el texto supera el ancho de la pestaña, podemos hacer que ocupe varias lineas.
- **Wdth and height …**: En caso de haber seleccionada un ancho fijo, aquí le indicamos el ancho y alto que queremos utilizar.
- **Orientation**: Posición en la que queremos que aparezcan las pestañas.

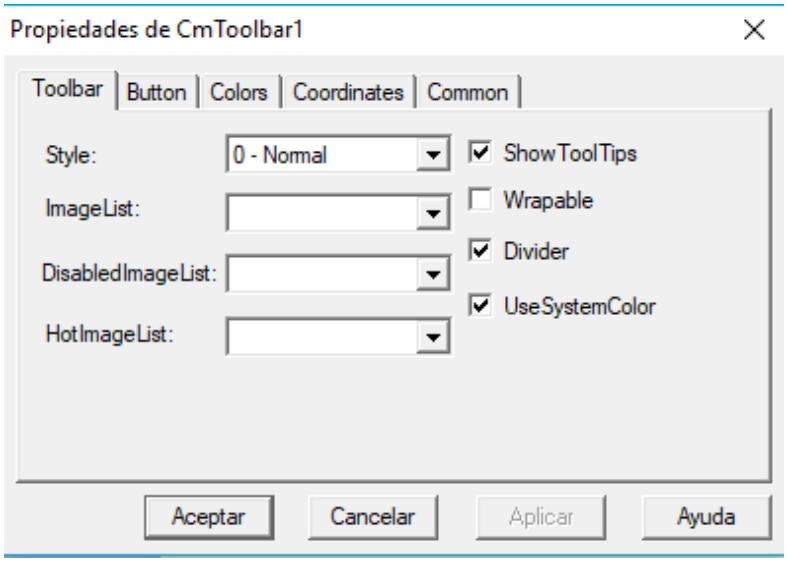
Pestaña Page:

- **Current Page**: Aquí seleccionamos el texto que aparecerá en cada una de las pestañas, indicamos el número y, en **Caption**, introducimos el texto, teniendo en cuenta que, la palabra que contenga el símbolo “***&***”, será la que usemos como acceso rápido a esa pestaña al pulsar la tecla *Alt*, por ejemplo “*&Datos Principales*”, ésto es común para la configuración de **Caption** en todos los controles y en **ToolTipText**, introducimos el texto que queramos que aparezca cuando pasamos el ratón por encima del control, ésto también es común para todos los controles.



PowerCobol ToolBar Control.

Inserta una barra de menú en nuestro form.



Pestaña Toolbar:

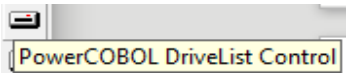
- **Style**: Indicamos el formato de barra de menú que queremos.
  - **0 – Normal**: La más común, con los botones en 3D.
  - **1 – Flat**: Plana, con los botones planos y el texto mostrado debajo de los botones.
  - **2 – FlaList**: Plana, con los botones planos y el texto mostrado a la derecha de los botones.
- **ImageList**: Nombre de la imagen incrustada que contiene los iconos de los botones.
- **DisableImageList**: Nombre de la imagen incrustada que contiene los iconos de los botones cuando están inhabilitados.
- **HotImageList**: Nombre de la imagen incrustada que contiene los iconos de los botones, cuando están pulsados.
- **ShowToolTips**: Indicamos si queremos que se muestre la ayuda flotante de los botones.
- **Wrapable**: Si hay más botones que ancha es la pantalla, indicamos si los queremos en más de una linea o no.
- **Divider**: Indicamos si queremos los separadores, (lineas verticales entre botones), o no.
- **UseSystemColor**: Le indicamos que utilice los colores por defecto de Windows, si lo marcamos, se ignorará cualquier cambio que hagamos en la pestaña *Colors*. Si no lo marcamos, el color del botón, será el que seleccionemos en la opción *BackColor* de la pestaña *Colors* y el texto con el que pongamos en la opción *ForeColor*, el color de resaltado y de sombreado, será definido por el sistema.

Pestaña Button:

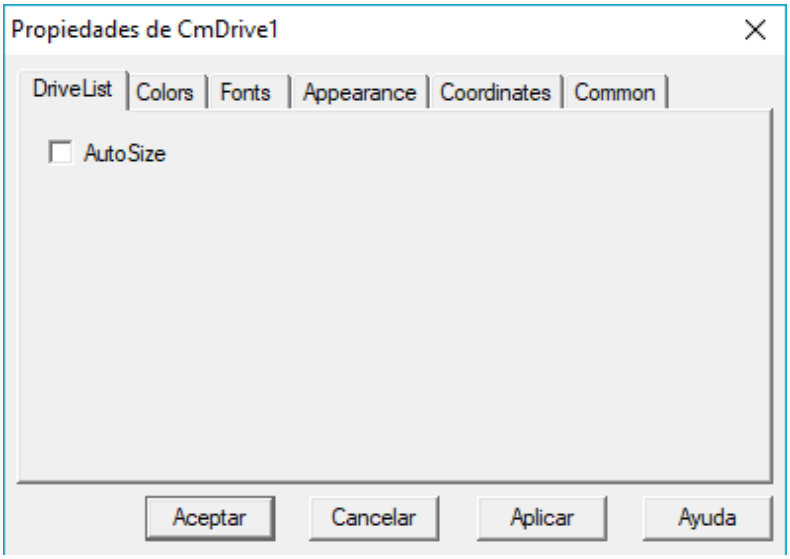
- **Index**: Es el número de elemento que vamos a modificar. (Elemento, porque los separadores, también cuentan), Si es la primera vez que accedemos a esta opción, estará inhabilitada y con el número 1 hasta que pulsemos en el botón A*dd*.
- **Style**: Es el formato que queremos para el elemento enumerado en *Index*.
  - **0 – Normal**: Un botón normal y clásico.
  - **1 – Separator**: Separador entre botones.
  - **2 – Check Button**: Botón especial que se queda pulsado hasta que volvamos a hacer click sobre él.
  - **3 - Check Button Group**: Un grupo de botones especiales del tipo *Check Button*, con la diferencia que, para que uno deje de estar “presionado”, hay que presionar en otro del grupo.
  - **4 – DropDown Button**: Botón con una punta de flecha hacia abajo, cuando pulsamos éste botón, se genera un evento *DropDown*, en el que podemos mostrar un menú del tipo *popup*, (desplegable flotante), por ejemplo.
- **Caption**: Texto para el botón.
- **ToolTipText**: Texto de la Ayuda Flotante para ese botón.

- **ImageIndex**: De la “tira” que contiene las imágenes de los botones, qué número de imagen es la que corresponde al botón.
- **Refer**: Muestra las imágenes de la “tira” de imágenes, para que seleccionemos la adecuada.
- **Visible**: Indica si el botón va a ser visible o no.
- **Enabled**: Indica si el botón va estar habilitado o no, en caso de que tengamos algo especificado en Disable*ImageList*, en la pestaña *Toolbar*, ése será el icono que muestre por defecto.
- **Value**: Indica si el botón estará “pulsado” por defecto, en caso de que tengamos algo especificado en *HotImageList*, en la pestaña *Toolbar*, ése será el icono que muestre por defecto.
- **Add**: Añade un elemento más al final del todo.
- **Insert**: Inserta un elemento a la derecha del que estemos en ese momento. (el que ponga en el apartado *Index*).
- **Delete*:*** *Elimina el elemento enumerado en el apartado Index*.

PowerCobol DriverList Control.



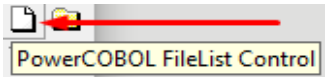
Inserta un control para ver el árbol de discos y carpetas de nuestro ordenador. Por defecto, muestra siempre el disco *C:*, de nuestro ordenador, y no se puede cambiar en las propiedades del control, sólo por programación. Se utiliza en combinación con los controles *FileList* y *FolderList*



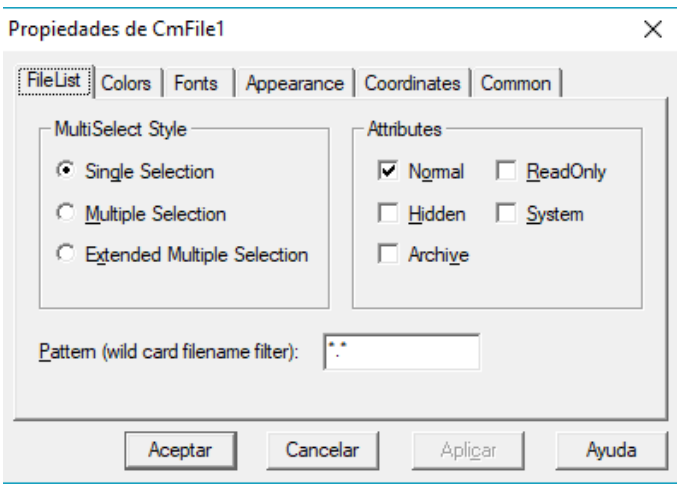
Pestaña DriverList:

- **AutoSize**: El tamaño del desplegable, intenta ajustarse automáticamente al número de elementos a mostrar.

PowerCobol FileList Control.



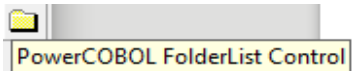
Inserta un control que muestra los ficheros que hay dentro de una carpeta, Se utiliza en combinación con los controles *DriverList* y *FolderList*.



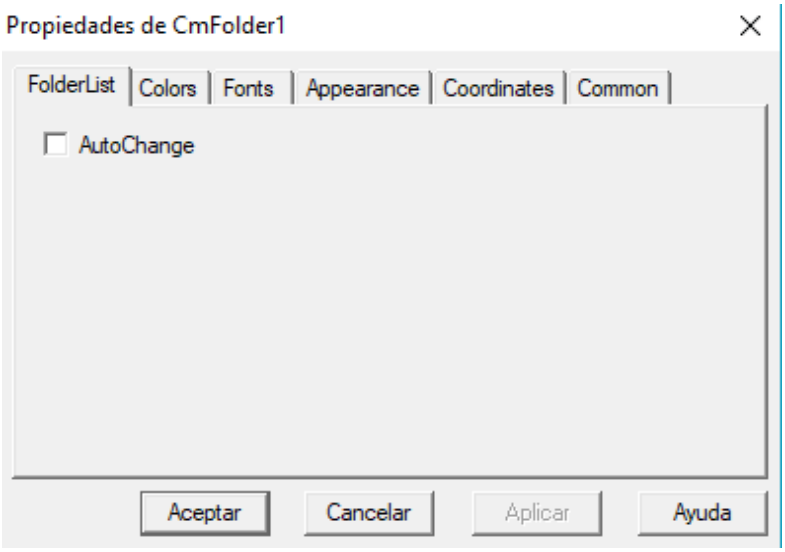
Pestaña FileList:

- **MultiSelect Style**:
  - **Single Select**: Sólo podremos seleccionar un elemento.
  - **Multiple Select**: Permite seleccionar más de un elemento. (Con la tecla Ctrl.)
  - **Extended Select**: Igual que el anterior, pero permite seleccionar también entre rangos. (Con las teclas Alt+Ctrl)
- **Attributes**: Indicamos qué tipos de archivos queremos mostrar.
- **Pattern (wild card …)**: Especifica la cadena que vamos a usar para que coincida con los nombres de los ficheros que queremos mostrar.

PowerCobol FolderList Control.



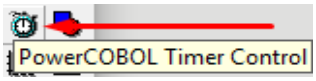
Inserta un control que muestra las carpetas que hay dentro de una unidad o disco duro, Se utiliza en combinación con los controles *DriverList* y *FileList*.



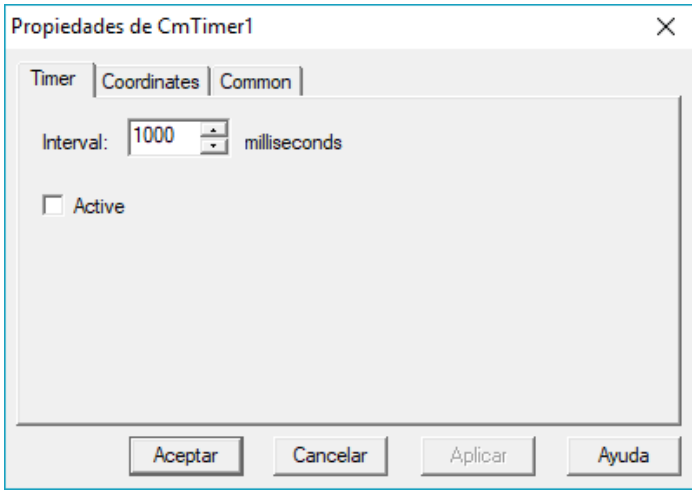
Pestaña FolderList:

- **AutoChange**: Si lo marcamos, hace que la carpeta en la posicionados, sea la carpeta por defecto, pero ésto no es notificado a los demás controles que dependan de éste.

PowerCobol Timer Control.



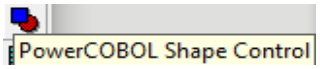
Inserta un control de ejecución cada nn tiempo en nuestro form. (Como el que se usaría para mostrar un reloj, o una ventana con un contador regresivo, etc.).



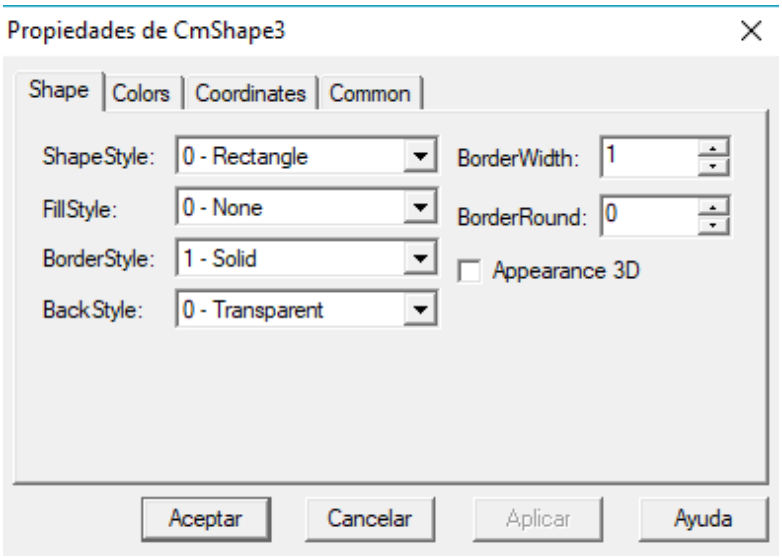
Pestaña Timer:

- **Interval**: Determina cada cuantos milisegundos se ha de ejecutar el control, el valor máximo es 65535 milisegundos, (un minuto y cinco segundos).
- **Active**: Indica si cuando ejecutemos el programa, el timer estará en ejecución.

PowerCobol Shape Control.

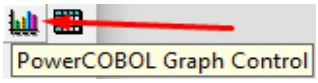


Inserta una forma geométrica, (cuadrado, rectángulo, elipse, …), en nuestro form.



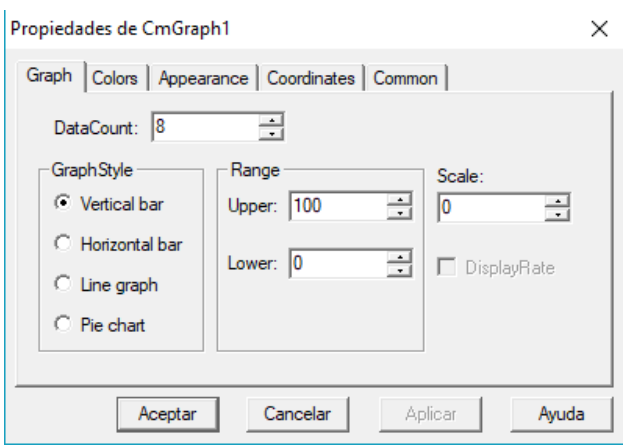
Pestaña Shape:

- **ShapeStyle**: *Rectangle*/*Square*/*Circle*/*Ellipse*, podemos elegir entre rectángulo, cuadrado, círculo o elipse.
- **FillStyle**: Indicamos como queremos que sea el relleno de la figura.
- **BorderStyle**: Indicamos que tipo de borde queremos.
- **BackStyle**: *Transparent*/*Opaque*/*HalfTransparent*, podemos elegir que el fondo sea transparente, opaco o semitransparente.
- **Appearance 3D**: Apariencia 3D o plana, hay que tener en cuenta que si elegimos en BorderRound un valor superios a 0, (cero), la apariencia 3D no tendrá efecto.
- **BorderWidth**: Ancho del borde de la figura
- **BorderRound**: redondeo de los bordes, sólo es aplicable cuando la figura es un rectángulo o un cuadrado.



PowerCobol Graph Control.

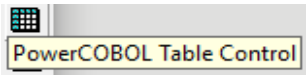
Inserta un gráfico en nuestro form.



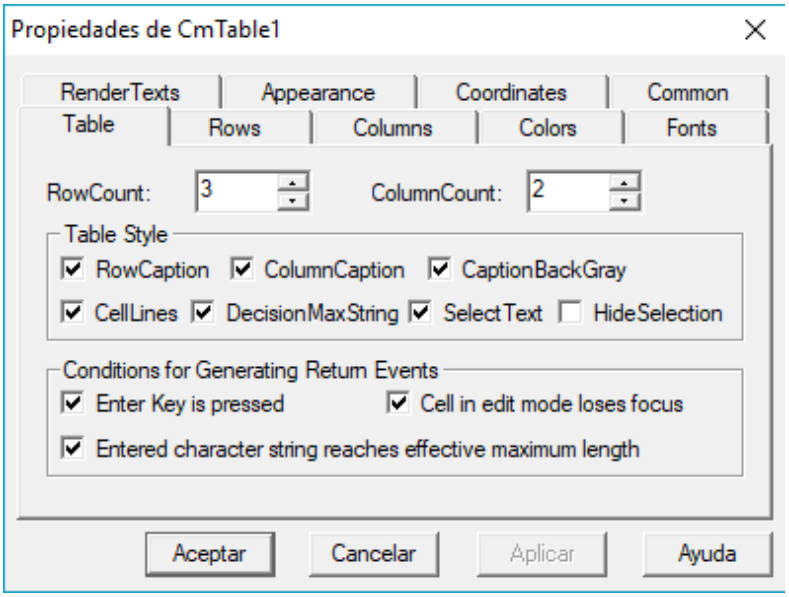
Pestaña Graph:

- **DataCount**: Indica el número de “columnas” que tendrá nuestro gráfico.
- **GraphStyle**: Tipo de gráfico.
  - **0 - Vertical Bar**: Gráfico Vertical.
  - **1 - Horizontal Bar**: Gráfico Horizontal.
  - **2 - Line graph**: Gráfico de lineas.
  - **3 - Pie Chart**: Gráfico circular o de tarta.
- **Upper**: Valor máximo de la gráfica.
- **Lower**; Valor mínimo de la gráfica.
- **Scale**: Indica el intervalo en la escala de graduación entre el máximo y el mínimo. Excepto para el gráfico de tarta, en el que no tiene efecto.
- **DisplayRate**: Muestra el porcentaje, (%), en números de cada sección del gráfico, (sólo para el gráfico de tarta).

PowerCobol Table Control.



Inserta una tabla en nuestro form. (Tipo a una hoja de cálculo). Es parecido al control *listView*, aunque con otras opciones y totalmente distintas de programar. Por ejemplo una tabla no se puede ordenar automáticamente, pero sí se puede cambiar el color celda por celda. Aunque la más importante es que, una tabla no acepta más de 2.000 lineas.



Pestaña Table:

- **RowCount**: Número de filas que tendrá por defecto. (Al contrario que en el control *listView*, aquí podemos definir cuantas filas tendrá nuestra tabla nada más empezar).
- **ColumnCount**: Número de columnas. (255 como máximo).
- **TableStyle**: Definimos el formato de la tabla.
  - **RowCaption**; Indica si queremos que las filas vayan numeradas.
  - **ColumnCaption**: Indica si queremos que las columnas vayan referenciadas. (A, B, C, D … FF).
  - **CellLines**: Indica si queremos ver la “rejilla” de la tabla.
  - **DecisionMaxString**: Determina si la entrada ha sido completada automáticamente cuando lleguemos al tamaño máximo que le hayamos dado en la **PICTURE** a esa columna.
  - **SelectText**: Cuando vayamos a modificar una celda, el texto se autoselecciona al hacer click en ella.
  - **HideSelection**: Si no lo marcamos, cuando hacemos click en una celda, ésta quedará remarcada con un pequeño rectángulo, para saber en que celda estamos, si lo marcamos, no aparecerá el rectángulo.
- **Conditions for Generating …**: En que condiciones se generará el evento *Retun*
  - **Enter Key is pressed**: Cuando pulsemos la tecla *Enter*/*Intro*.

- **Entered string reaches …**: Cuando el texto escrito llegue a la longitud máxima definida en la cláusula **PICTURE**.
- **Cell in edit mode …**; Cuando la celda que estamos modificando, pierda el foco. (Hagamos click en otro sitio).

Pestaña Rows:

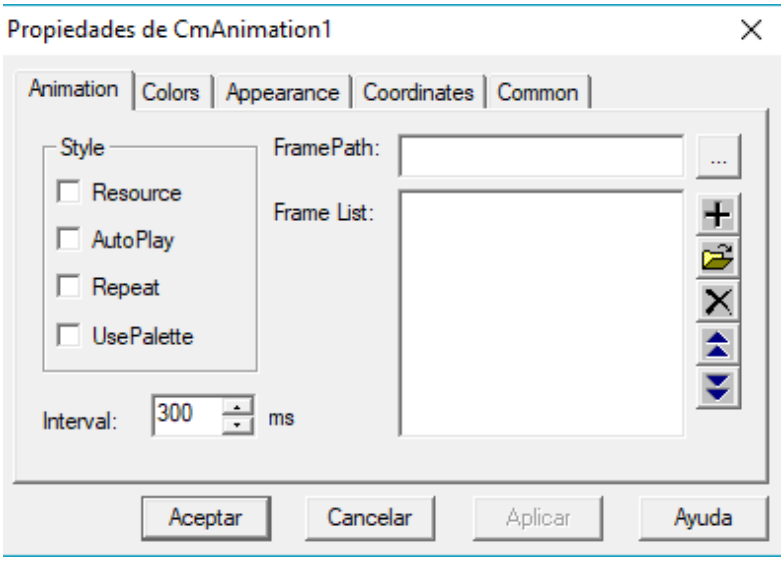
- **AutoRowHeight**: Altura automática de las celdas.
- **DefaultRowHeight**: Altura de las celdas definida por el número que pongamos.

Pestaña Columns:

- **Table Columns**: Muestra las columnas con sus nombres, con las flechas que tiene al lado, podemos cambiar el orden de las columnas. Automáticamente, aparecerán tantas columnas como hayamos definido en la propiedad *ColumnCount* de la pestaña *Table*. Para poder hacer cualquier cambio, primero tenemos que seleccionarlo en ésta tabla.
- **Caption**: Es el texto que queremos que aparezca en la cabecera de la columna.
- **Width**: Ancho que queremos para la columna.
- **Alignment**: Alineación del texto.
- **IMEMode**: Aconsejo no tocarlo, se utiliza para uso interno de Windows y como trata los datos.
- **Writable**: Hace que toda la columna no se pueda modificar ningún dato.
- **ScrollLock**: Desde esa columna hacia la izquierda, quedan fijas, por lo que, aunque usemos la barra de desplazamiento horizontal, siempre estarán visibles.
- **WordWrap**: Si el texto es más ancho que la columna, hace un salto de linea o no lo hace.
- **AutoEdit**: Indica si la celda se ha de poner en modo edición en cuanto recibe el foco. (hacemos click sobre ella),

PowerCobol Animation Control.

Inserta una sucesión de imágenes animadas en nuestro form. (Tipo a un gif animado).

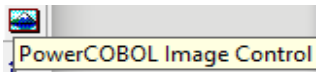


Pestaña Animation:

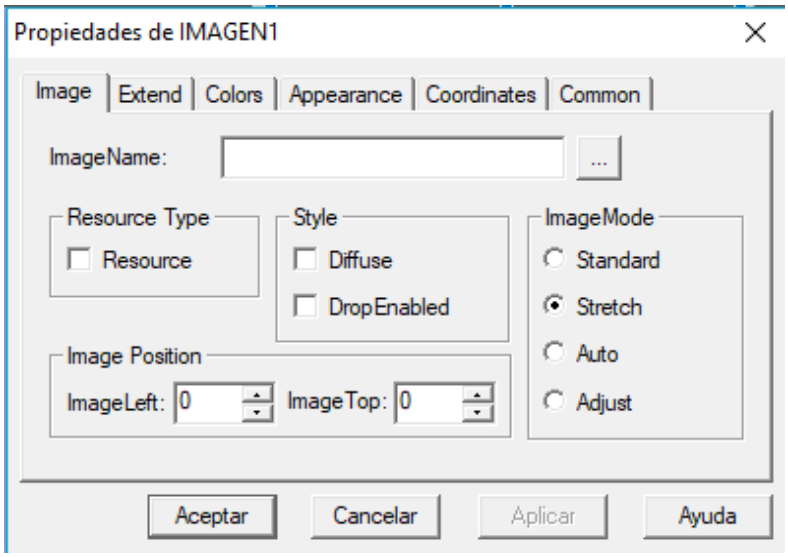
- **Style:**
  - **Resorce**: Lo marcamos si vamos a utilizar imágenes incrustadas en el programa en vez de imágenes de nuestro disco duro.
  - **Autoplay**: Si lo marcamos, la sucesión de imágenes comenzará nada más abrir el programa.
  - **Repeat**: Cuando termina, vuelve a empezar automáticamente.
  - **UsePalette**: Al marcarlo, si la paleta de colores de la imagen supera a la que puede manejar PWC, se intentará ajustar los colores lo mejor posible, ésto puede hacer que se ralentice la presentación. Si no lo marcamos, y hay más colores, puede que las imágenes no se vean correctamente.
- **Interval**: Intervalo de tiempo entre las imágenes en milisegundos, desde 0 hasta 65.535.
- **FramePath**: En caso de no usar imágenes incrustadas, aquí indicamos la ruta donde está la sucesión de imágenes. Si no vamos a usar imágenes incrustadas, éste campo es **obligatorio**.

- **FrameList**: En ésta ventana, tenemos las imágenes en el orden en el que se van a presentar. Con los botones de la derecha, podemos añadir, eliminar, y/o cambiar de posición cualquier imagen.

PowerCobol Image Control.



Inserta una imagen en nuestro form.



Pestaña Image:

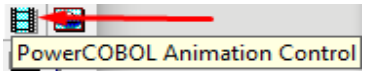
(Todas las opciones de ésta pestaña, se pueden cambiar en tiempo de ejecución).

- **ImageName**; Aquí indicaremos el nombre completo, (incluida la ruta), de la imagen.
- **Resource**: Indica si la imagen va a ser un recurso, (una imagen incrustada en el programa), con lo que en *ImageName*, pondremos el nombre del que le asignemos al recurso, o si no lo marcamos, la imagen vendrá de un fichero en nuestro equipo. Como más adelante vamos a incrustar imágenes en el programa, sobre todo para los botones, lo veremos con más detalle.
- **Style**:
  - **Diffuse**: Si lo marcamos, usará unos algoritmos especiales para dibujar la imagen en el caso de que la seleccionada, exceda en resolución y colores a la soportada por PWC, es útil, pero como contrapartida, si hay muchas imágenes, el tiempo de “pintado” aumenta considerablemente.
  - **DropEnabled**: Indica que la imagen mostrada, puede ser “Arrastrada y Soltada” en otro sitio o no, Ésto sólo es efectivo, si la opción *Resource*, está como *False*, (no seleccionada).
- **ImageMode**: *Standard*/*Stretch*/*Auto*/*Adjust*, le indicamos el tipo de ajuste para la imagen, es más rápido probar cada una de las opciones, que explicarlas.
- **Image Position**: *ImageLeft*/*ImageTop*, posición de la imagen contando con que (0,0) es la esquina superior izquierda

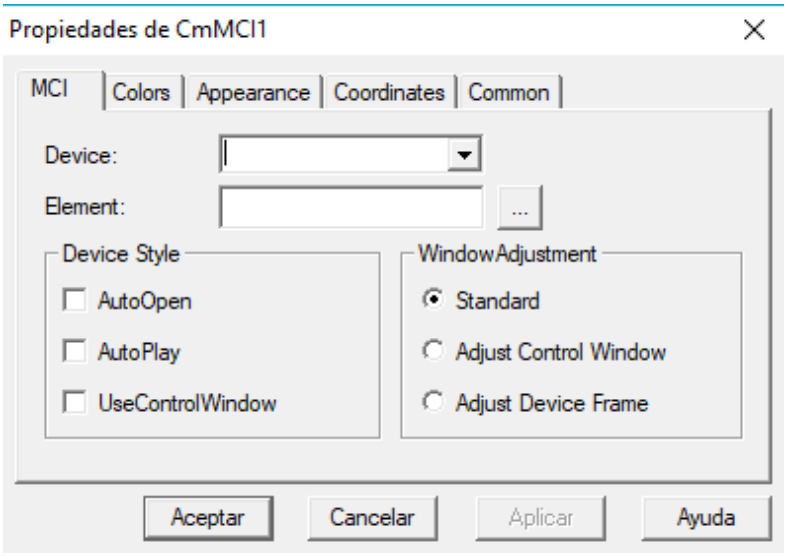
Pestaña Extend:

Ésta sólo la vamos a utilizar si por una casualidad, el formato de imagen que vamos a utilizar es **PCD**, (Kodak Photo CD ®)

PowerCobol MCI Control.



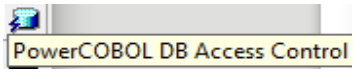
Inserta un control Multimedia en nuestro form.



Pestaña MCI:

- **Device**: Especificamos el formato del archivo Multimedia que vamos a reproducir.
  - **0 - AVIVideo**: Vídeo en formato AVI.
  - **1 - CDAudio**: CD-Rom de música.
  - **2 - MPEGVideo**: Vídeo en formato MPEG
  - **3 - Sequencer**: Audio para sequenciadores. (MIDI)
  - **4 - WaveAudio**: Audio en formato WAV.
- **Element**: Indicamos o buscamos, (con el botón de la derecha), el fichero a reproducir.
- **Device Style**:
  - **AutoOpen**: Indicamos que el dispositivo se va a abrir automáticamente, nada más iniciar el programa.
  - **AutoPlay**: Indicamos que el fichero se va a ejecutar automáticamente, nada más iniciar el programa.
  - **UseControlWindow**: Indicamos que abra el reproductor predeterminado de Windows.
- **Windows Adjustment**: Ajustes de la ventana de reproducción.
  - **Standard**: Sin ajustes.
  - **Adjust Control Window**: Ajusta la imagen para que quepa en la ventana.
  - **Adjust Device Frame**: Ajusta la ventana al tamaño de la imagen. (Sólo es efectivo si está marcada la opción *UseControlWindow*).

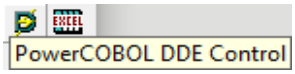
PowerCobol DB Access Control.



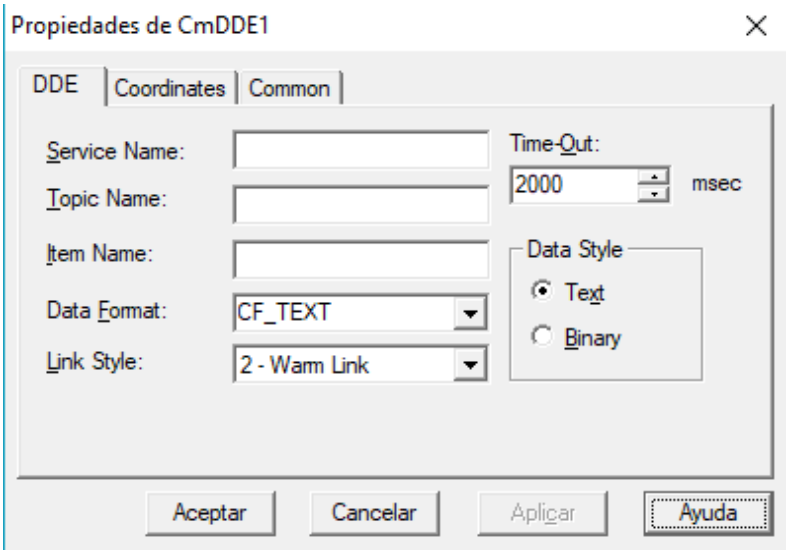
Inserta un control de acceso a Bases de Datos en nuestro form.

No lo he utilizado nunca, por lo que, para no meter la pata, lo voy a pasar por alto. Si alguien quiere enviarme una descripción de su uso, al final del manual hay una dirección de contacto para ello.

PowerCobol DDE Control.

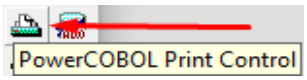


Inserta un control DDE en nuestro form. (Un control que permite ejecutar ficheros .EXE con parámetros)

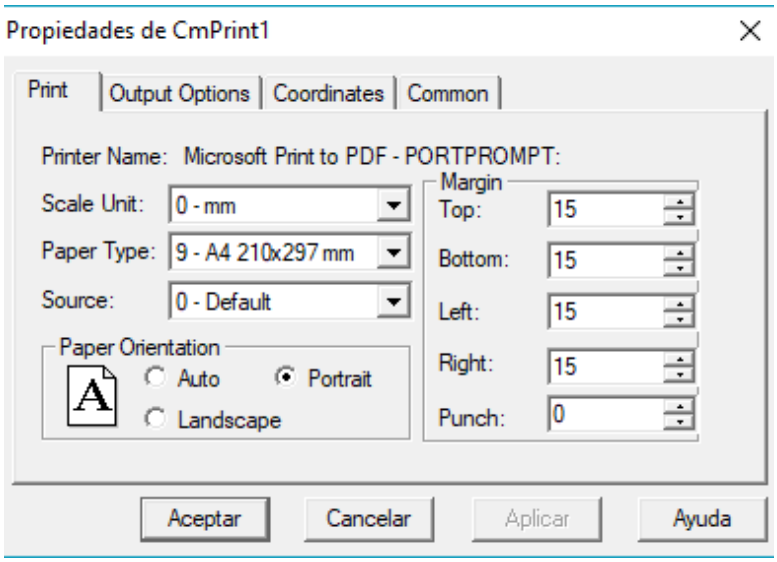


Nunca he configurado éstos parámetros, siempre que lo he usado, ha sido desde programación con una simple instrucción `INVOKE CmDDE1 "Execute" USING ejecutable RETURNING RETORNO` y me ha funcionado perfectamente. Sé que se puede configurar el ejecutable, la ruta, los parámetros de vuelta, formato de archivos, etc. Así que digo lo mismo que en anterior, si alguien lo ha usado más profundamente y quiere remitirme la información, no hay problema.

PowerCobol Print Control.

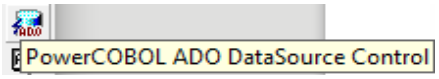


Inserta un control de impresora en nuestro form. (Se utiliza únicamente para hacer un volcado de pantalla a la impresora, no es necesario para imprimir un listado, por ejemplo).

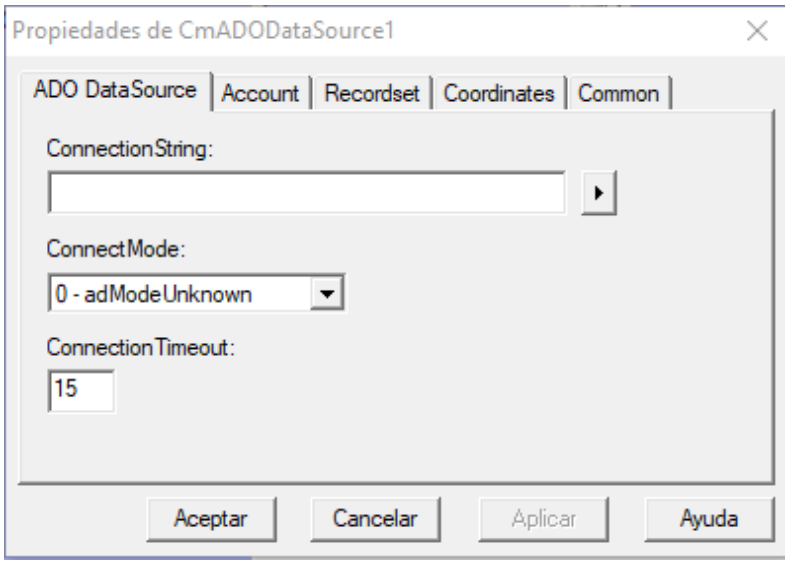


Se configura exactamente igual que configuraríamos una impresora.

PowerCobol ADO DataSource Control.



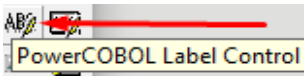
Inserta un control ADO de conexión a Bases de Datos. (**A**ctiveX **D**ata **O**bjects).



Pestaña ADO DataSoure:

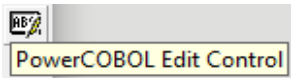
- **ConenectionString**: Es la cadena de texto que vamos a usar para conectarnos a la DB, puede ser una cadena de conexión o el nombre que hemos definido en la configuración de ODBC.
- **ConnectMode**: Es el modo de conexión. (Desconocida, sólo lectura, sólo escritura, compartida, etc.)
- **ConnectionTimeout**: Es el tiempo que debe de estar probando el control a conectarse, hasta que de un error si no lo consigue.

PowerCobol Label Control.



Inserta un control Label en nuestro form. (un texto sin opciones). Es parecido al control *Static Text*, pero sin opciones, yo no lo he usado nunca.

PowerCobol Edit Control.



No lo he utilizado nunca, y tampoco sé para que sirve exactamente, por lo que, para no meter la pata, lo voy a pasar por alto. Si alguien quiere enviarme una descripción de su uso, al final del manual hay una dirección de contacto para ello.

Pestañas comunes a todos los controles >>

Pestaña Colors:

Ya la hemos explicado al principio de éste documento.

Pestaña Coordinates:

En ésta pestaña, es donde damos las coordenadas del control, así como la escala y si se puede redimensionar.

- **ScaleMode**; Indica la unidad de medida de todos los componentes, yo uso siempre *Píxeles*, ya que es la unidad con la que se configuran las pantallas.
- **Left/Top**: Indica la posición del control, tomando como comienzo la esquina superior izquierda del mismo. La medida toma como referencia la esquina superior izquierda del form o, del control en el que lo insertemos, (un *GroupBox,* un *tabControl* o un *frameBox*).
- **Width/Height**: Alto/Ancho del control, si lo modificamos desde aquí, se "ensancha" por la parte de la derecha y se "alarga" por la parte de abajo. Nunca al revés.
- **ScalingStyle**: Si redimensionamos la ventana, se puede elegir que redimensionamos. Yo lo tengo deshabilitado siempre, porque ésto no es el punto fuerte de PWC, falla mucho y queda horrible.

Pestaña Appearance:

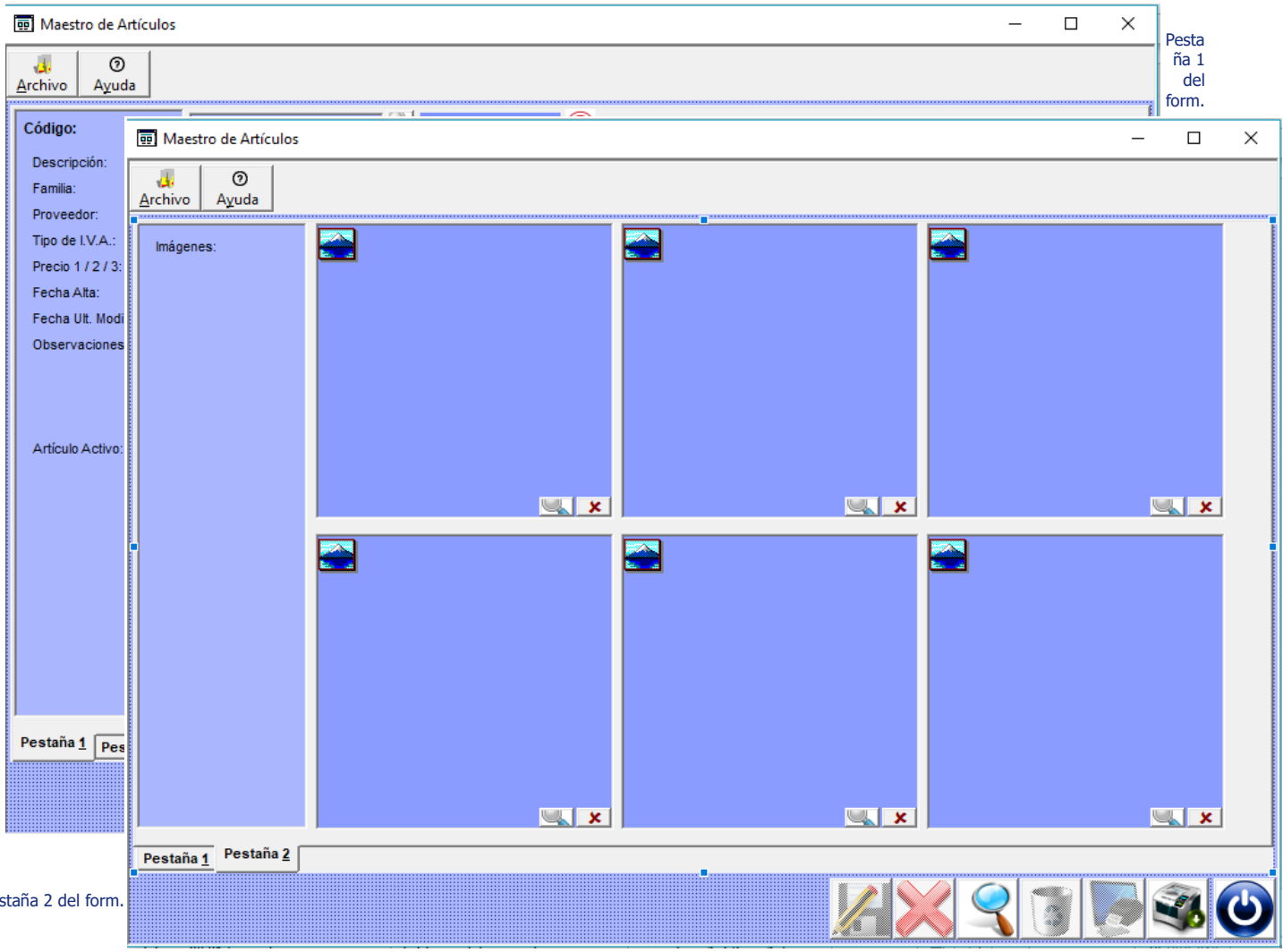
- **BackStyle**: Pude ser Transparent, el control muestra el color de fondo, (es decir sobre dónde está situado el control, (el form, un *groupBox*, un *frameBox* ...), u Opaque, el control muestra el color que definamos en la pestaña Colors.
- **BorderStyle**: *Off* o *Solid*, no muestra ningún borde o si lo muestra.
- **Appearance**: *Flat/3D*, cambia la apariencia entre plana o 3D
- **Enabled**; Si está marcado, lo muestra en color normal, si está desmarcado, lo muestra en un tono gris que indica que está deshabilitado.

Pestaña Common: (ésto es aplicable a todos los controles)

- **Name**:Aquí nos aparecerá el nombre que PWC le pone por defecto al control, este nombre es con el que nos referiremos al control a la hora de escribir el código fuente del programa, no se puede cambiar en tiempo de ejecución y hay que procurar que sea un nombre más o menos descriptivo del control, y no poner cosas raras como por ejemplo "asifkjji", que luego no vamos a saber qué es ni nos vamos a acordar del nombre, yo en mi caso, utilizo "Campo1, Campo2, … Campo*n*" para los que hay que introducir textos, etc.
- **Visible**: indica si el campo se verá o estará oculto.
- **TabStop**: Indica si el campo recibirá "el foco" cuando pulsemos la tecla *Tab*, se puede cambiar en tiempo de ejecición.
- **TabGroup**: Determina si el control, será el primero en el grupo de *Tabs*, nunca he tenido que usar esta propiedad, así que …
- **TabIndex**: Aquí le indicamos el orden en el que el foco, se moverá de un control a otro al pulsar la tecla *Tab*, no se puede cambiar en tiempo de ejecución.
- **Array**: Se utiliza para crear una tabla indexada de controles, lo que equivaldría a una definición OCCURS en COBOL, lo explicaré con un ejemplo más adelante.
- **Index**: En caso de haber creado un *Array*, es el número de elemento, (índice), dentro del array.
- **ToolTipText**: Es el pequeño texto descriptivo que aparece en todos los programas cuando, por ejemplo, dejamos el ratón quieto en un botón. El tamaño máximo es 8192 caracteres, tener en cuenta, que no soporta saltos de linea, es bastante simple y feo, pero es lo que hay.

<< Otras acciones >>

Vamos a crear todos los campos del form, pestañas, texto, editables, imágenes, botones, barras, etc., pero recordar que el primero que tenemos que crear es el control *Tab*, (TabControl), dentro del cual, van a ir la mayoría de controles que tenemos que crear, si no lo hacemos así, luego no podremos mover ningún objeto creado fuera de éste control al interior del mismo, por lo que nos tocará borrar los controles y volverlos a crear.
Yo una vez terminado el form, se me ha quedado de un tamaño de 928x640 píxeles, no tiene por qué coincidir con el vuestro.



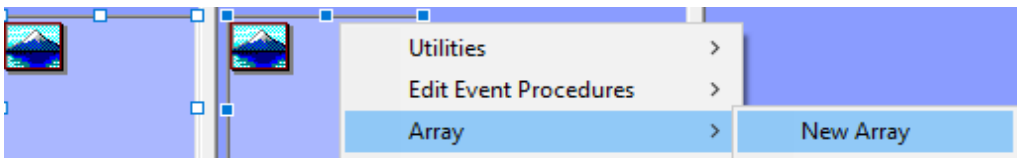
Pestaña 1 del form.

Pestaña 2 del form.

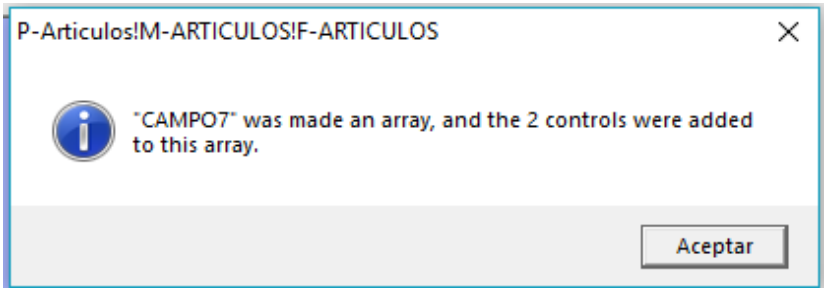
Crear un Array:

Si os fijáis, en la pestaña 2, hay 6 controles de imágenes, 6 botones con una lupa y otros 6 con una **X**. Aunque no es necesario ni imprescindible para éste proyecto, vamos a crear un **Array** de campos, (es algo parecido a definir un campo en COBOL y asignarle la cláusula **OCCURS**), con cada uno de los grupos de éstos elementos.

Para ello, hacemos click con el botón izquierdo en la primera imagen de la linea superior, después pulsamos la tecla *Ctrl* y sin soltarla, vamos haciendo click con el botón izquierdo en cada una de las demás imágenes en orden, (ésto es muy importante, porque ese será el orden en el que nos creará el índice del array, y si nos equivocamos o queremos cambiar el orden, nos tocará eliminar el array y volver a crearlo). si no hemos programado ningún código aún, no hay problemas, de lo contrario, todo lo programado se perderá. Una vez seleccionados las 6 imágenes, hacemos click con el botón derecho en la primera de la izquierda y, en el menú desplegable, seleccionamos *Array → New Array*



Nos aparecerá una ventana indicándonos que *nombre_de_control*, se va a convertir en un array y que *nn* controles van a ser añadidos al array, aceptamos ...



Ejemplo de ventana de pregunta

… y si nos fijamos en el nombre del campo, (arriba a la derecha, si está seleccionado), éste habrá cambiado de *nombre_de_control* a *nombre_de_control(1)*, el de la derecha será *nombre_de_control(2)* … hasta el último que será *nombre_de_control(6)*.

Ahora, tenemos que hacer exactamente lo mismo con los botones de las lupas y después con los botones de las "**X**", y con ello, se os quedará más o menos como a mi.

Vamos con la BASE de DATOS.

Para repasar un poco cómo se programan Bases de Datos, (en adelante BD), el fichero de artículos va a ser una tabla en una BD. PWC, permite el acceso a BDs via ADO, (ActiveX Data Objects), vía DB, (parecido a como si programásemos en Access de Microsoft ®), o vía SQL Embebido, que es el que vamos a utilizar en este manual.

SQL Embebido acepta muchas de las instrucciones nativas de SQL, pero tiene ciertas limitaciones en cuanto a estas, por ejemplo no acepta instrucciones CREATE, DROP o TRUNCATE, por lo que para crear la BD y las tablas, o lo hacemos antes desde un gestor de SQL, (HEIDI, XAMPP, WAMPP, etc.), o nos toca llamar al sistema para crearlas de manera externa, (con un control *DDE* o una instrucción tipo `INVOKE "ExecuteSync" USING parámetros` lo mismo pasa para borrar y/o vaciar una tabla y algunas más que no recuerdo ahora.

Crear la BD

Si no tenéis la BD creada, os adjunto el código para crear la que vamos a utilizar en este manual: (si por un casual, esta o cualquier otra instrucción SQL, no os funciona u os da un error, cambiar la comilla «`» por ésta otra comilla simple «‘», no sé por qué razón unos gestores aceptan una y otros gestores aceptan la otra). De todas formas, junto a éste manual, se adjuntarán todos los recursos necesarios, (iconos, imágenes, copy's, fuentes, etc.)

```
CREATE DATABASE IF NOT EXISTS `manual_cobol`
```

y para crear la tabla, utilizaremos las siguientes sentencias SQL:

```
CREATE TABLE IF NO EXIST `articulos` (
`ARTCOD` VARCHAR(15) NOT NULL COLLATE 'latin1_spanish_ci',
`ARTDES` VARCHAR(60) NULL DEFAULT NULL COLLATE 'latin1_spanish_ci',
`ARTFAM` VARCHAR(3) NULL DEFAULT NULL COLLATE 'latin1_spanish_ci',
`ARTPRO` DECIMAL(6,0) UNSIGNED NOT NULL DEFAULT '0',
`ARTIVA` DECIMAL(5,2) UNSIGNED NOT NULL DEFAULT '0',
`ARTPRE1` DECIMAL(10,3) UNSIGNED NOT NULL DEFAULT '0',
`ARTPRE2` DECIMAL(10,3) UNSIGNED NOT NULL DEFAULT '0',
`ARTPRE3` DECIMAL(10,3) UNSIGNED NOT NULL DEFAULT '0',
`ARTOBS` VARCHAR(200) NULL DEFAULT NULL COLLATE 'latin1_spanish_ci',
`ARTACT` VARCHAR(1) NULL DEFAULT 'S' COLLATE 'latin1_spanish_ci',
`ARTIMG1` VARCHAR(255) NULL DEFAULT NULL COLLATE 'latin1_spanish_ci',
`ARTIMG2` VARCHAR(255) NULL DEFAULT NULL COLLATE 'latin1_spanish_ci',
`ARTIMG3` VARCHAR(255) NULL DEFAULT NULL COLLATE 'latin1_spanish_ci',
`ARTIMG4` VARCHAR(255) NULL DEFAULT NULL COLLATE 'latin1_spanish_ci',
`ARTIMG5` VARCHAR(255) NULL DEFAULT NULL COLLATE 'latin1_spanish_ci',
`ARTIMG6` VARCHAR(255) NULL DEFAULT NULL COLLATE 'latin1_spanish_ci',
`ARTFEC` VARCHAR(8) NULL DEFAULT NULL COLLATE 'latin1_spanish_ci',
`ARTFEC1` VARCHAR(8) NULL DEFAULT NULL COLLATE 'latin1_spanish_ci',
`ARTELI` VARCHAR(1) NULL DEFAULT 'N' COLLATE 'latin1_spanish_ci',
PRIMARY KEY (`ARTCOD`),
INDEX `ARTFAM` (`ARTFAM`)
)
COMMENT='Tabla de Artículos'
COLLATE='latin1_spanish_ci'
ENGINE=InnoDB
;
```

con eso ya tenemos creada la BD y la tabla que vamos a utilizar.

Enlazar la BD con PWC. (ODBC, .INF y COBOL85.cbr):

ODBC:

Con SQL Embebido, necesitamos crear un enlace entre nuestro programa y la BD vía ODBC, así que vamos a ello, abrimos el gestor de ODBCs, (normalmente estará en Windows, *Panel de Control → Herramientas Administrativas → Orígenes de datos ODBC*).hacemos click en *Agregar*, seleccionamos *Maria ODBC n.n Driver* y hacemos click en *Finalizar*.

1. En *Name*, vamos a poner **ManCOBOL**, es el nombre con el que nos referiremos más adelante en otra configuración que tendremos que realizar.
2. En *Description*, vamos a poner **Manual de COBOL**, es una descripción, así que dará lo mismo lo que pongamos.
3. Hacemos click en *Siguiente* o *Next*.
4. Seleccionamos *TCP/IP*
5. En *Server Name*: ponemos **127.0.0.1** ó **localhost**. Aunque siempre es mejor la dirección IP, que el nombre.
6. En *Port**,*** ponemos **3306**, (Si no lo hemos modificado cuando instalamos MySQL o MariaDB, en cuyo caso tendremos que poner el puerto que indicamos en su momento)
7. En *User Name*, ponemos un usuario que tenga permisos suficientes para leer/grabar en la BD, (normalmente *root*).
8. En *Password*, ponemos la contraseña para ese usuario en concreto.
9. Pulsamos el el botón *Test DSN*, (a la derecha de Password).
10. Si todo ha sido correcto, nos lo indicará en una ventana emergente, en caso contrario, algo hemos hecho mal, por lo que habrá que repasar desde el principio).
11. Hacemos click en el desplegable *Database*, y seleccionamos la nuestra **manual_cobol**.
12. Hacemos click en *Siguiente* o *Next*.
13. Marcamos *Enable automatic reconnect*, y el resto los desmarcamos si fuera necesario..
14. Hacemos click en *Siguiente* o *Next*..
15. Volvemos a hacer click en *Siguiente* o *Next*.
16. Y otra vez más, hacemos click en *Siguiente* o *Next*.
17. Y para finalizar, hacemos click en *Finalizar* o *Finish*.

.INF:

Ahora vamos a crear el archivo **.inf**, que PWC necesita para saber dónde está el conector y sus parámetros, para ello, vamos a la carpeta de instalación de PWC, (por defecto ***C:\Program Files (x86)\Fujitsu NetCOBOL for Windows***), y buscamos un programa que se llama **SQLODBCS.exe** y, si estamos en Windows 7 o superior, hacemos click con el botón derecho y seleccionamos *Ejecutar como Administrador, (muy importante, aunque seamos administradores del sistema)*, para Windows XP, lo ejecutamos sin más.

1. Como el fichero **.inf**, no existe aún, en la ventana que se nos ha abierto, pulsamos en *Browse*, y navegamos por las carpetas hasta llegar a donde vamos a tener el programa instalado, que si recordáis del principio de éste manual, es **C:\Articulos**, y en la parte de abajo, en *Nombre del fichero*, vamos a poner **BD.inf**, y hacemos click en *Abrir*
2. Volvemos a la ventana inicial y hacemos click en*Ok*. Como he dicho antes, el fichero no existe, por lo que nos saldrán un par de ventanas de advertencia, a la primera hacemos click en *Yes* y a la segunda en *Accept*.
3. En *SERVER NAME*, ponemos el nombre que queramos, yo en mi caso **JOSBER**.
4. Seleccionamos *Machine Data Source*
5. En el desplegable *Data Source Name*, seleccionamos **ManCOBOL**.
6. En *User ID*, seleccionamos un usuario que pueda conectarse a la BD con todos los permisos, igual que antes.
7. En *Password*, introducimos la contraseña de ese usuario.
8. En *Comment*, ponemos Manual de COBOL, aunque igual que antes, es un simple comentario.
9. Seleccionamos *Read-Write*.
10. Seleccionamos *Auto* en *Commit Mode*.
11. Hacemos click en *Aplicar* o Apply.
12. *Hacemos click en Aceptar* o *Accept*.

COBOL85.cbr:

Por último, sólo nos queda configurar el archivo **COBOL85.CBR**, para indicarle a PWC, dónde está el fichero **.inf**, para ello, vamos a la carpeta de instalación de PWC, (por defecto ***C:\Program Files (x86)\Fujitsu NetCOBOL for Windows***), y buscamos un programa que se llama **COBENVUT.exe** y, si estamos en Windows 7 o superior, hacemos click con el botón derecho y seleccionamos *Ejecutar como Administrador, (muy importante, aunque seamos administradores del sistema)*, para Windows XP, lo ejecutamos sin más.

1. Como el fichero **COBOL85.CBR**, aún no existe, en la ventana que se nos ha abierto, seleccionamos del menú, *File → Open*, (arriba a la izquierda), y navegamos por las carpetas hasta llegar a donde vamos a tener el programa instalado, que si recordáis del principio de éste manual, es **C:\Articulos**, y en la parte de abajo, en *Nombre del fichero*, vamos a poner **COBOL85.CBR**, y hacemos click en *Abrir*
2. Seleccionamos Single Thread.
3. En el desplegable *Variable Name*, buscamos la que se llama @*ODBC_Inf* y la seleccionamos.
4. En *Variable Value*, hacemos click en el botón que tiene a la derecha, buscamos el archivo que hemos creado y lo seleccionamos.
5. Una vez seleccionado, hacemos click en el botón *Set*, que está a la derecha
6. Hacemos click en el bottón *Apply*.
7. Y ya podemos cerrar la ventana.

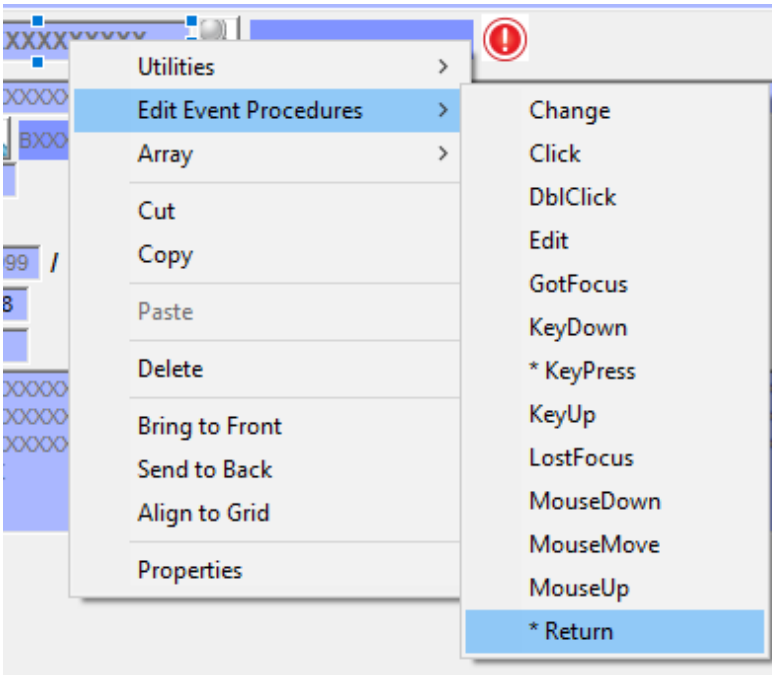
Y si todo lo hemos hecho bien, ya podemos conectarnos desde el programa sin ningún problema, aunque eso es algo que de momento no podremos saber, por lo menos hasta que hayamos programado algo de código fuente.

Y por fin … vamos con la programación.

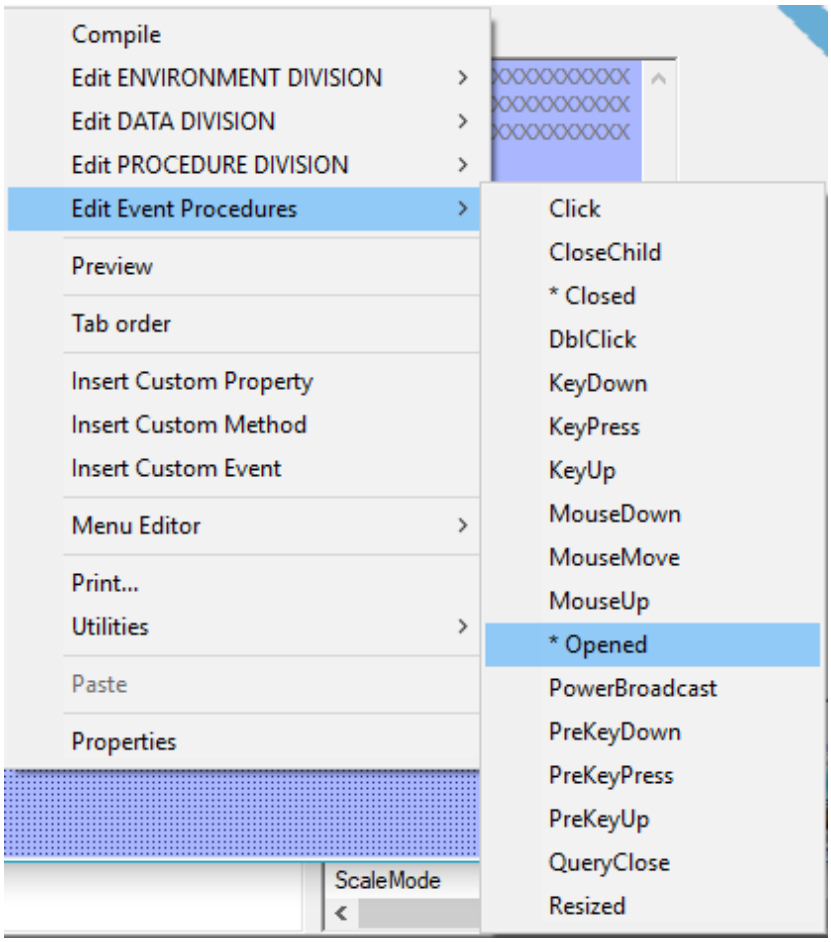
<< Consideraciones antes de empezar >>

Antes de empezar a programar y, para que luego no nos volvamos locos preguntándonos por qué “leshes” no funciona nuestro programa, o por qué no hace lo que nosotros queremos, hay que tener en cuenta algunas *cosillas* bastante importantes:

1. Para los que venís aún de un entorno de programación en modo texto, (que seréis pocos, pero alguno habrá aún), la programación **orientada a objetos** ó ***OO***, implica que cada objeto, (botón, texto, tabla, etc.) es por decirlo de alguna manera un objeto independiente de los demás, por lo que cada uno tendrá sus programas completos e independientes de los otros, que ese programa podrá tener 2 lineas o miles de ellas, pero, y ésto es seguro, tendrá todas sus *DIVISION, SECTION, ETIQETAS* , *EXIT PROGRAM, etc.*
2. Además, cada vez que el usuario final, pulsa una tecla, (da lo mismo que sea una “A” que F7),o hace click en un botón, o entra a editar un campo, cualquier cosa que se os ocurra, que pueda hacerse u ocurrir en un formulario, ésto genera lo que se conoce como un **Evento**. Al igual que os he dicho antes, cada evento, es un pequeño programa totalmente completo, que ejecutará un determinado código, que realizará una determinada acción, desde grabar un registro, imprimir una ficha o terminar el programa y, al igual que antes, serán totalmente independientes unos de otros. (otra cosa es que se puedan llamar entre ellos, como en cualquier lenguaje de programación).
3. Por una parte, tenemos la programación del form, lo que sería la “ventana común” que contiene dentro de ella, todos los demás elementos, como botones, campos, tablas, etc. Todos los form, tienen una parte que es común a los objetos que los componen, lo que quiere decir que es “heredado” por esos objetos, (por lo que no hay que repetirla en esos objetos), estas son la **ENVIRONMENT DIVISION** y **DATA DIVISION**, con todas sus **SECTION**, etc., (excepto **PROCEDURE DIVISION**, que, aunque parezca increíble, no conozco a nadie que la haya usado o sepa para qué se usa, porque, por mucho código que programes ahí, no sé cuando ni como se ejecuta), y los subprogramas o subrutinas que no están relacionados con los eventos antes descritos, (lo que aquí se llama *New procedure*).
4. Si pulsáis con el botón derecho en cualquier elemento del form, excepto en el propio form, el desplegable que se abre es similar para todos ello, en él hay un apartado que se llama *Edit Event Procedures*, que, dependiendo del objeto, habrán unos eventos u otros, (los que empiezan con un asterisco, (*****), es porque ya tienen algún código programado, por lo que a partir de ahora, cuando me refiera con un, por ejemplo, *vamos al evento Return del Campo1*, significa que tenéis que hacer click con el botón derecho en el objeto ***Campo1*** → *Edit Event Procedures* → *Return*.



Todos los form tienen también sus propios evento, los más comunes de usar son *Opened*, que es el que se ejecuta nada más iniciar el programa, el evento *Closed*, que es el último en ejecutarse antes de que el programa se cierre y el evento *CloseChild*, que es el que se ejecuta cuando se cierra un programa al que hemos llamado desde el form en el que estamos. A estos eventos, se accede haciendo click con el botón derecho, en cualquier parte del form en la que no haya ningún elemento, seleccionando *Edit Event Procedures y ahí seleccionamos el evento que nos interesa.*



Nuestro programa:

El programa que vamos a hacer es un simple mantenimiento de un fichero de artículos, en el que introducimos el código del artículo en el campo1 y al pulsar *Enter*, pueden ocurrir 2 cosas,

1: que no exista, con lo que “limpiamos de datos” las 2 pestañas, inhabilitamos los botones y menús que no nos hagan falta, para evitar que pulsen en ellos, y pasamos a crearlo o ...

2: que exista, con lo que cargamos todos los campos y entramos en modificación,

además, vamos a tener en cuenta que, si introducimos un texto en el código, y pulsamos las teclas *Ctrl. + B* o hacemos click en el botón de búsqueda por código, (la lupa pequeña), haremos una consulta de todos los artículos en los que el código contenga el texto introducido, (o todos los artículos de la tabla si no se ha introducido ningún texto en el código). El programa nos permitirá eliminar y hacer una consulta del fichero, listar o hacer un volcado completo de la pantalla hacia la impresora, para lo que incluiremos un control *Print* en nuestro form.

También podremos crear o consultar familias en el mismo momento, para evitar el tener que salir de los artículos si una familia no existe o no sabemos cuál es, por lo que controlaremos si, desde CAMPO*3*, pulsamos la tecla *Ctrl.+A,* con lo que “llamaríamos” al programa de familias, para realizar un **Alta**, o si pulsamos la tecla *Ctrl.+B*, con lo que “llamaríamos” al programa de **consulta de familias**, aunque también tenemos la pequeña lupa, que hay a la derecha de CAMPO3, que nos lleva también, al igual que pulsar la tecla *Ctrl.+B*, a la consulta de familias.

Y, como eso es todo, empecemos …

Programación del Form

Click con el botón derecho en el form y vamos a definir los datos comunes a todo el form

ENVIRONMENT DIVISION → SPECIAL-NAMES:

```
  DECIMAL-POINT IS COMMA.
```

ENVIRONMENT DIVISION → FILE CONTROL:

```
     SELECT OPTIONAL FAMILIAS
            ASSIGN TO "C:\Articulos\Datos\FAMILIAS.FIC"
            ORGANIZATION IS INDEXED
            ACCESS MODE  IS DYNAMIC
            RECORD KEY   IS FAMCOD
            FILE STATUS  IS STFAM
            LOCK IS AUTOMATIC.
```

DATA DIVISION → FILE SECTION:

```
 FD  FAMILIAS GLOBAL EXTERNAL
     LABEL RECORD IS STANDARD
     DATA RECORD IS REG-FAM.
 01  REG-FAM.
     03  FAMCOD               PIC XXX.
     03  FAMDES               PIC X(30).
```

DATA DIVISION → WORKING-STORAGE SECTION:

```
 01  ESTILO                   PIC S9(4) COMP-5   GLOBAL.
 01  RETORNO                  PIC S9(4) COMP-5   GLOBAL.
 01  WS-ATENCION              PIC X(17)          GLOBAL VALUE " ¡ Atención ... !".
 01  SALTO                    PIC X  VALUE X"0A" GLOBAL.
 01  AHIVA3T                  PIC XXX            GLOBAL EXTERNAL.
 01  AHIVA15T                 PIC X(15)          GLOBAL EXTERNAL.
*
 01  STFAM                    PIC XX             GLOBAL.
*
* --- * DECLARACIÓN DE LAS VARIABLES PARA SQL
*
                 EXEC SQL
                      BEGIN DECLARE SECTION
                 END-EXEC.
*
 01  REG-ART      GLOBAL.
     03  ARTCOD               PIC X(15).
     03  ARTDES               PIC X(60).
     03  ARTFAM               PIC XXX.
     03  ARTPRO               PIC S9(6).
     03  ARTIVA               PIC S999V99.
     03  ARTPRE1              PIC S9(7)V999.
     03  ARTPRE2              PIC S9(7)V999.
     03  ARTPRE3              PIC S9(7)V999.
     03  ARTOBS               PIC X(200).
     03  ARTACT               PIC X.
     03  ARTIMG1              PIC X(255).
     03  ARTIMG2              PIC X(255).
     03  ARTIMG3              PIC X(255).
     03  ARTIMG4              PIC X(255).
     03  ARTIMG5              PIC X(255).
     03  ARTIMG6              PIC X(255).
```

```
     03  ARTFEC              PIC X(8).
     03  ARTFEC1             PIC X(8).
     03  ARTELI              PIC X.      *> Para saber si un artículo ha sido
eliminado (S/N).
      *>
      *> --- VARIABLES
      *>
 01  SQLSTATE                PIC XXXXX   IS GLOBAL.
 01  SQLCODE                 PIC S9(9) COMP-5   GLOBAL.
 01  SQLMSG                  PIC X(600)     GLOBAL.
 01  CONTASQL                PIC S9(9)   IS GLOBAL.  *> CONTADOR DE NÚMERO DE
REGISTROS
*
           EXEC SQL
                END DECLARE SECTION
           END-EXEC.
*
  01  FECHA GLOBAL.
      03  AA                  PIC 9999.
      03  MM                  PIC 99.
      03  DD                  PIC 99.
  01  FECHA1 GLOBAL.
      03  DD                  PIC 99.
      03  MM                  PIC 99.
      03  AA                  PIC 9999.
```

Aquí, en la WORKING, vamos a definir las variables que necesitemos para todos los programas, por eso están definidas todas como GLOBAL y también las variables que vamos a utilizar en SQL. Cualquier cosa que vayamos a hacer con **SQL Embebido**, ha de empezar obligatoriamente con la sentencia `EXEC SQL` y terminar con `END-EXEC`, aparte, cuando definimos variables numéricas para usar con SQL, **es obligatorio** que lleven signo (`PIC S9`…), lo vayamos a usar o no, de lo contrario, nos dará un error al compilar.

Eventos del Form

Evento *Closed* del form:

Aquí vamos a cerrar el fichero de familias y desconectarnos de la BD.

```
 ENVIRONMENT     DIVISION.
  DATA            DIVISION.
  WORKING-STORAGE SECTION.
  PROCEDURE       DIVISION.

      CLOSE FAMILIAS.

      EXEC SQL
           DISCONNECT 'JOSBER'
      END-EXEC.

      INVOKE POW-SELF "CloseForm".
```

Evento *Opened* del form:

Aquí vamos a abrir el fichero de familias, a conectarnos a la BD y a ejecutar varios subprogramas. Si os fijáis en el código que pongo aquí debajo, donde nos conectamos a la BD, no pongo ni el nombre de la BD, ni de la conexión ODBC, si no, el nombre que pusimos cuando creamos el archivo **.inf**, tenerlo en cuenta, porque es un fallo muy común.

```
 ENVIRONMENT     DIVISION.
  DATA            DIVISION.
  WORKING-STORAGE SECTION.
  01  WTEX                    PIC X(700).
  PROCEDURE       DIVISION.

      OPEN INPUT FAMILIAS.

*
* ----- SQL
*
      EXEC SQL
           CONNECT TO 'JOSBER'
      END-EXEC.
*
      IF SQLSTATE NOT = "00000"
         ADD POW-DMICONERROR POW-DMOK GIVING ESTILO
         MOVE SPACES TO WTEX

         STRING "Error al conectar a la BD de MySQL"
                SALTO
                "["
                SQLSTATE
                "]"
                SALTO DELIMITED BY SIZE INTO WTEX
         END-STRING

         INVOKE POW-SELF "DisplayMessage"
                USING SQLMSG
                      SQLSTATE
                      ESTILO
         END-INVOKE

         INVOKE POW-SELF "CloseForm"
      END-IF.
*
      CALL "VACIAR".
      CALL "INHABILITAR".
      INVOKE CAMPO1 "SetFocus".
      EXIT PROGRAM.
```

Con éste código, nos conectamos a la BD, y comprobamos que no haya habido ningún error. Siempre que hagamos algo en la BD, MariaDB, nos va a devolver el estado de la acción en la variable **SQLSTATE**, ésta hay que definirla con ese nombre exacto en la **WORKING** de nuestro programa, porque PWC, la identifica y "sabe" que tiene que hacerlo así. Sería el equivalente a **FILE-STATUS**, de los ficheros COBOL.
Después, ejecutamos unas rutinas que lo que hacen es limpiar todos los campos de las pantallas e inhabilitar los que no necesitemos en el momento, para evitar, por ejemplo, que pulsen en un botón que no deben. Acto seguido, le pasamos el foco al campo1 y salimos del programa. Recordar que, siempre que paséis el foco a un componente, (un campo, un botón, etc.), la siguiente instrucción que ha de aparecer es **EXIT PROGRAM** o el foco no se irá a ese componente, si no que continuará con la siguiente instrucción que haya, es un error muy común también, y que al principio hará que os volváis locos, mirando a ver porque el programa "*no se va al botón*" que le habéis indicado.

Programación de los Controles

Pestaña 1

Evento *keypress* del campo1:

El evento *keypress*, se ejecuta a cada tecla pulsada, da lo mismo la que sea, así que si escribimos la palabra *hola* se ejecutará el evento 4 veces.

```
 ENVIRONMENT     DIVISION.
  DATA            DIVISION.
  WORKING-STORAGE SECTION.
  01   I                        PIC 99 BINARY.
  01   WTEX                     PIC X(50).
  LINKAGE         SECTION.
  01   POW-ARG-KEYASCII PIC S9(4) COMP-5.
  PROCEDURE       DIVISION USING POW-ARG-KEYASCII.
*
      EVALUATE POW-ARG-KEYASCII
               WHEN 27   *> Tecla Escape
                    CALL "PARA-SALIR"
               WHEN 02   *> Tecla Ctrl. + B
                    IF "Text" OF CAMPO1 = SPACES
                       MOVE 1 TO "CurrentPage" OF TAB1
                       INVOKE CAMPO1 "SetFocus"
                       EXIT PROGRAM
                    END-IF

                    MOVE "Text" OF CAMPO1 TO AHIVA15T
                    INVOKE POW-SELF "CallForm" USING "F-CON-ARTICULOS"
                    MOVE AHIVA15T TO "Text" OF CAMPO1
                    MOVE AHIVA15T TO ARTCOD

                    IF AHIVA15T = SPACES
                       MOVE 1 TO "CurrentPage" OF TAB1
                       INVOKE CAMPO1 "SetFocus"
                       EXIT PROGRAM
                    END-IF

                    EXEC SQL
                         SELECT * FROM ARTICULOS
                                WHERE BINARY ARTCOD = :ARTCOD
                         INTO :ARTCOD,
                              :ARTDES,
                              :ARTFAM,
                              :ARTPRO,
                              :ARTIVA,
                              :ARTPRE1,
                              :ARTPRE2,
                              :ARTPRE3,
                              :ARTOBS,
                              :ARTACT,
```

```
                              :ARTIMG1,
                              :ARTIMG2,
                              :ARTIMG3,
                              :ARTIMG4,
                              :ARTIMG5,
                              :ARTIMG6,
                              :ARTFEC,
                              :ARTFEC1,
                              :ARTELI
                       END-EXEC

                       IF SQLSTATE NOT = "00000"    *> Ha habido un error
                          MOVE SPACES TO WTEX
                          COMPUTE I = FUNCTION STORED-CHAR-LENGTH (ARTCOD)
                          STRING "El artículo "
                                 ARTCOD (1:I)
                                 ", no existe ..." DELIMITED BY SIZE INTO WTEX
                          END-STRING
                          ADD POW-DMICONERROR POW-DMOK GIVING ESTILO
                          INVOKE POW-SELF "DisplayMessage"
                                 USING WTEX
                                       WS-ATENCION
                                       ESTILO
                          END-INVOKE
                          CALL "INHABILITAR"
                          CALL "VACIAR"
                          MOVE 1 TO "CurrentPage" OF TAB1
                          INVOKE CAMPO1 "SetFocus"
                          EXIT PROGRAM
                       ELSE
                          CALL "HABILITAR"
                          CALL "VACIAR"
                          CALL "EXISTE"
                          MOVE 1 TO "CurrentPage" OF TAB1
                          INVOKE CAMPO2 "SetFocus"
                          EXIT PROGRAM
                       END-IF
           END-EVALUATE.
```

En éste código, comprobamos las teclas, si pulsamos *Esc*, nos salimos del programa, y si pulsamos *Ctrl+B*, hacemos una consulta, y al volver, comprobamos si nos devuelve algún artículo seleccionado, si es que no, volvemos al campo1, que es que sí, leemos el artículo del fichero, lo mostramos en pantalla, (`CALL "EXISTE"`), habilitamos todos los campos y botones necesarios, (`CALL "HABILITAR"`) y damos el foco al CAMPO2.

Y ahora, un par de cosas sobre SQL Embebido, la primera es, procurar no introducir comentarios dentro de una instrucción SQL, puede no pasar nada, o puede tener errores imprevistos, (que es lo más normal), ni aunque los pongáis con el *, (asterisco), en la columna 7, y la segunda es que, cuando hagáis referencia a una variable dentro de una instrucción SQL, la variable ha de empezar irremediablemente con **:**, (dos puntos), seguido del nombre de la variable, (por ejemplo :ARTCOD, :CONTASQL, etc.), y esto no es opcional.

Evento *Return* del campo1

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.
*
     IF "Text" OF CAMPO1 = SPACES
        INVOKE CAMPO1 "SetFocus"
        EXIT PROGRAM
     END-IF.
*
     MOVE "Text" OF CAMPO1 TO ARTCOD.

*
* --- * con la cláusula BINARY, MariaDB distingue entre mayúsculas y minúsculas
*
```

```
     EXEC SQL
          SELECT * FROM ARTICULOS
                 WHERE BINARY ARTCOD = :ARTCOD
                 INTO :ARTCOD,
                      :ARTDES,
                      :ARTFAM,
                      :ARTPRO,
                      :ARTIVA,
                      :ARTPRE1,
                      :ARTPRE2,
                      :ARTPRE3,
                      :ARTOBS,
                      :ARTACT,
                      :ARTIMG1,
                      :ARTIMG2,
                      :ARTIMG3,
                      :ARTIMG4,
                      :ARTIMG5,
                      :ARTIMG6,
                      :ARTFEC,
                      :ARTFEC1,
                      :ARTELI
     END-EXEC.
*
     CALL "HABILITAR".
*
     IF SQLSTATE NOT = "00000"
        CALL "NO-EXISTE"
        INVOKE CAMPO2 "SetFocus"
        EXIT PROGRAM
     END-IF.
*
     CALL "EXISTE".
     INVOKE CAMPO2 "SetFocus".
     EXIT PROGRAM.
```

Básicamente, el código es muy parecido al del evento *KeyPress*

Evento *KeyPress* de los campos 2, 3, 4, 9, 10 y 11.

```
 ENVIRONMENT     DIVISION.
  DATA            DIVISION.
  WORKING-STORAGE SECTION.
  LINKAGE         SECTION.
  01   POW-ARG-KEYASCII PIC S9(4) COMP-5.
  PROCEDURE       DIVISION USING POW-ARG-KEYASCII.
*
     EVALUATE POW-ARG-KEYASCII
              WHEN 27   *> Tecla Escape
                   CALL "PARA-SALIR"
     END-EVALUATE.
```

Evento KeyPress de los campos 5, 6, 7 y 8.

```
 ENVIRONMENT     DIVISION.
  DATA            DIVISION.
  WORKING-STORAGE SECTION.
  LINKAGE         SECTION.
  01   POW-ARG-KEYASCII PIC S9(4) COMP-5.
  PROCEDURE       DIVISION USING POW-ARG-KEYASCII.
*
     EVALUATE POW-ARG-KEYASCII
              WHEN 27   *> Tecla Escape
                   CALL "PARA-SALIR"

              WHEN 46   *> Cambia el . del teclado numérico por una ,
```

```
                    MOVE 44 TO POW-ARG-KEYASCII

     END-EVALUATE.
```

La diferencia con el anterior es que, si pulsamos el **.**, (punto) del teclado numérico, éste se cambia por una **,** (coma), esto es así, para hacer más cómoda la introducción de números con decimales.

Evento *Return* de los campos 2, 4, 5, 6, 7, 8, 9 y 10

```
 ENVIRONMENT     DIVISION.
  DATA            DIVISION.
  WORKING-STORAGE SECTION.
  PROCEDURE       DIVISION.

     INVOKE CAMPO3 "SetFocus".
```

Aquí lo que hacemos es que, al pulsar *Return*, vayamos al campos siguiente, la diferencia entre ellos es a que campo le damos el foco, en el campo2 es al campo3, pero en el campo3, será al campo4 … No he puesto la instrucción **`EXIT PROGRAM`**, después de **`INVOKE xxxxx "SetFocus"`**, porque al ser la última instrucción, no es necesario.

Evento *KeyPress* del campo3

Este se detalla, porque en campo3, podemos crear una familia, (si pulsamos *Ctrl.+A*), consultar una familia, (si pulsamos *Ctrl.+B*), o comprobar si existe la familia, (si pulsamos *Enter*/*Intro*).

```
 ENVIRONMENT     DIVISION.
  DATA            DIVISION.
  WORKING-STORAGE SECTION.
  LINKAGE         SECTION.
  01  POW-ARG-KEYASCII PIC S9(4) COMP-5.
  PROCEDURE       DIVISION USING POW-ARG-KEYASCII.
*
     EVALUATE POW-ARG-KEYASCII
              WHEN 01   *> Ctrl.+A (Vamos a realizar un Alta de Familia)
                   IF "Text" OF CAMPO3 = SPACES
                      MOVE SPACES TO AHIVA3T
                   ELSE
                      MOVE "Text" OF CAMPO3 TO AHIVA3T
                   END-IF

                   CLOSE FAMILIAS
                   INVOKE POW-SELF "CallForm" USING "F-FAMILIAS" "M-FAMILIAS"
                   OPEN INPUT FAMILIAS
                   MOVE AHIVA3T TO "Text" OF CAMPO3

                   IF AHIVA3T = SPACES
                      INVOKE CAMPO3 "SetFocus"
                      EXIT PROGRAM
                   ELSE
                      MOVE AHIVA3T TO FAMCOD
                      READ FAMILIAS NO LOCK
                           INVALID KEY
                                   MOVE " No existe ..." TO "Caption" OF INFORMACION1
                                   INVOKE CAMPO3 "SetFocus"
                                   EXIT PROGRAM
                           NOT INVALID KEY
                               MOVE FAMDES TO "Caption" OF INFORMACION1
                      END-READ
                      INVOKE CAMPO4 "SetFocus"
                      EXIT PROGRAM
                   END-IF

              WHEN 02   *> Ctrl.+B (Vamos a realizar una Consulta de Familia)
                   IF "Text" OF CAMPO3 = SPACES
```

```
                MOVE SPACES TO AHIVA3T
             ELSE
                MOVE "Text" OF CAMPO3 TO AHIVA3T
             END-IF

             INVOKE POW-SELF "CallForm" USING "F-CON-FAMILIAS" "M-FAMILIAS"
             MOVE AHIVA3T TO "Text" OF CAMPO3

             IF AHIVA3T = SPACES
                INVOKE CAMPO3 "SetFocus"
                EXIT PROGRAM
             ELSE
                MOVE AHIVA3T TO FAMCOD
                READ FAMILIAS NO LOCK
                     INVALID KEY
                             MOVE " No existe ..." TO "Caption" OF INFORMACION1
                             INVOKE CAMPO3 "SetFocus"
                             EXIT PROGRAM
                     NOT INVALID KEY
                         MOVE FAMDES TO "Caption" OF INFORMACION1
                END-READ
                INVOKE CAMPO4 "SetFocus"
                EXIT PROGRAM
             END-IF

          WHEN 27   *> Tecla Escape
             CALL "PARA-SALIR"

   END-EVALUATE.
```

Evento *Return* del campo3.

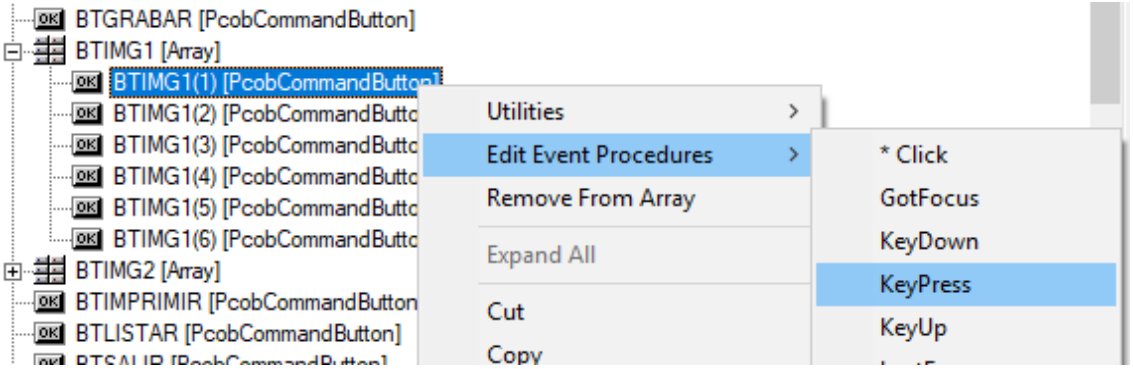
```
ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.

     CALL "MOSTRAR-FAM".
     INVOKE CAMPO4 "SetFocus".
```

Pestaña 2

Eventos en un Array de Controles

Cuando codificamos cualquier evento de un control que se ha convertido previamente en un array, lo que codificamos es para todos los controles de ese array, para eso tenemos siempre definida la variable **POW-ARG-INDEX** en la **LINKAGE SECTION** de todos los controles convertidos en array, ésta variable es la que contiene el número de índice dentro del array. Si queremos codificar únicamente para un sólo control, debemos hacerlo desde el árbol de elementos de la izquierda.



No va ser nuestro caso, así que vamos con los eventos, está programado para que, si hacemos doble-click en una imagen, se nos abra una ventana en la que podemos seleccionar una imagen desde nuestro disco duro.

Evento *DblClick* en Imagen1_n:

```
 ENVIRONMENT     DIVISION.
  DATA            DIVISION.
  WORKING-STORAGE SECTION.
*
  01  WTEX                     PIC X(255).
  LINKAGE         SECTION.
  01  POW-ARG-INDEX   PIC S9(9) COMP-5.
*
  PROCEDURE       DIVISION USING POW-ARG-INDEX.

      MOVE SPACES TO WTEX.

      INVOKE POW-SELF "GetFileName"
             USING WTEX
                   "Seleccione el Archivo"
      "*.jpg (Imagen JPG,JPE)|*.JPG;JPE|*.BMP (Imagen de Mapa de Bits)|*.tif|*.TIF
(Imagen TIFF)|*.tif|*.PNG (Portable Graphics Network)|*.png"  *>|*.* (Todos los
Archivos)|*.*"
                   POW-CDOPEN
             RETURNING RETORNO
      END-INVOKE.

      IF WTEX = SPACES
         INVOKE IMAGEN1 (POW-ARG-INDEX) "SetFocus"
         EXIT PROGRAM
      END-IF.

      MOVE WTEX TO "ImageName"   OF IMAGEN1 (POW-ARG-INDEX).
      MOVE WTEX TO "ToolTipText" OF IMAGEN1 (POW-ARG-INDEX).
      INVOKE IMAGEN1 (POW-ARG-INDEX) "Refresh".
```

Evento *Click* en BtImg1_n:

```
 ENVIRONMENT     DIVISION.
  DATA            DIVISION.
  WORKING-STORAGE SECTION.
*
  01  WTEX                     PIC X(255).
  LINKAGE         SECTION.
  01  POW-ARG-INDEX   PIC S9(9) COMP-5.
*
  PROCEDURE       DIVISION USING POW-ARG-INDEX.

      MOVE SPACES TO WTEX.

      INVOKE POW-SELF "GetFileName"
             USING WTEX
                   "Seleccione el Archivo"
      "*.jpg (Imagen JPG,JPE)|*.JPG;JPE|*.BMP (Imagen de Mapa de Bits)|*.tif|*.TIF
(Imagen TIFF)|*.tif|*.PNG (Portable Graphics Network)|*.png"  *>|*.* (Todos los
Archivos)|*.*"
                   POW-CDOPEN
             RETURNING RETORNO
      END-INVOKE.

      IF WTEX = SPACES
         INVOKE BTIMG1 (POW-ARG-INDEX) "SetFocus"
         EXIT PROGRAM
      END-IF.

      MOVE WTEX TO "ImageName"   OF IMAGEN1 (POW-ARG-INDEX).
      MOVE WTEX TO "ToolTipText" OF IMAGEN1 (POW-ARG-INDEX).
```

```
     INVOKE IMAGEN1 (POW-ARG-INDEX) "Refresh".
```

Evento *Click* en BtImg2_n:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  I                        PIC 9.
 01  WTEX                     PIC X(30).
 LINKAGE         SECTION.
 01  POW-ARG-INDEX   PIC S9(9) COMP-5.
 PROCEDURE       DIVISION USING POW-ARG-INDEX.
*
     IF "ImageName" OF IMAGEN1 (POW-ARG-INDEX) = SPACES
        EXIT PROGRAM
     END-IF.
*
     MOVE POW-ARG-INDEX TO I.
     MOVE SPACES TO WTEX.

     STRING "¿Borrar la imagen número "
            I
            "?" DELIMITED BY SIZE INTO WTEX
     END-STRING.

     ADD POW-DMICONQUESTION POW-DMYESNO GIVING ESTILO.
     INVOKE POW-SELF "DisplayMessage"
            USING WTEX
                  WS-ATENCION
                  ESTILO
            RETURNING RETORNO
     END-INVOKE.

     IF RETORNO = POW-DMRNO
        EXIT PROGRAM
     END-IF.

     MOVE SPACES TO "ImageName"   OF IMAGEN1 (POW-ARG-INDEX).
     MOVE SPACES TO "ToolTipText" OF IMAGEN1 (POW-ARG-INDEX).
     INVOKE IMAGEN1 (POW-ARG-INDEX) "Refresh".
     EXIT PROGRAM.
```

La penúltima instrucción que hay en los 3 eventos, `INVOKE IMAGEN1 (POW-ARG-INDEX) "Refresh".`, se utiliza para actualizar el control o componente, (una imagen, un listview, una tabla, ...), si no lo hacemos, se queda con el valor que tiene, en éste caso mostrando una imagen y no nos muestra los cambios que hemos hecho, por lo que es aconsejable utilizar ésta instrucción.

ToolBar

Los elementos de un *ToolBar*, se enumeran según el valor que tengan en la propiedad *Index*, de la pestaña *Button*, de las propiedades de la barra, por lo que si hemos puesto separadores, éstos también cuentan como un elemento.

Al igual que en un array, nos tenemos que referir a un elemento, por su número de índice, que ya nos viene definido en la **LINKAGE SECTION**, como **POW-ARG-BUTTONINDEX**.

Evento *ButtonClick* de CmToolbar1:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 LINKAGE         SECTION.
 01  POW-ARG-BUTTONINDEX PIC S9(9) COMP-5.
 PROCEDURE       DIVISION USING POW-ARG-BUTTONINDEX.

     EVALUATE POW-ARG-BUTTONINDEX
```

```
            WHEN 1
                 CLOSE FAMILIAS
                 MOVE SPACES TO AHIVA3T
                 INVOKE POW-SELF "CallForm" USING "F-FAMILIAS" "M-FAMILIAS"
                 OPEN INPUT FAMILIAS
           WHEN 3
                 INVOKE POW-SELF "CallForm" USING "F-ACERCA" "M-ACERCA"
     END-EVALUATE.
```

Botones inferiores del Form

Evento *Click* del botón BTGRABAR:

```
 ENVIRONMENT     DIVISION.
  DATA            DIVISION.
  WORKING-STORAGE SECTION.
  PROCEDURE       DIVISION.
*
     IF "Text" OF CAMPO1 = SPACES
        MOVE 1 TO "CurrentPage" OF TAB1
        INVOKE TAB1 "SetFocus"
        INVOKE CAMPO1 "SetFocus"
        EXIT PROGRAM
     END-IF.

     MOVE "Text" OF CAMPO1 TO ARTCOD.
     MOVE "Text" OF CAMPO2 TO ARTDES.
     MOVE "Text" OF CAMPO3 TO ARTFAM.
     MOVE "Text" OF CAMPO4 TO ARTPRO.
     MOVE "Text" OF CAMPO5 TO ARTIVA.
     MOVE "Text" OF CAMPO6 TO ARTPRE1.
     MOVE "Text" OF CAMPO7 TO ARTPRE2.
     MOVE "Text" OF CAMPO8 TO ARTPRE3.

     MOVE "Text" OF CAMPO9 TO FECHA1.
     MOVE CORR FECHA1 TO FECHA.
     MOVE FECHA TO ARTFEC.

     MOVE FUNCTION CURRENT-DATE TO ARTFEC1.

     MOVE "Text" OF CAMPO11 TO ARTOBS.

     IF "Value" OF CHECK1 = POW-TRISTATE-CHECKED
        MOVE "S" TO ARTACT
     ELSE
        MOVE "N" TO ARTACT
     END-IF.

     MOVE "ImageName" OF IMAGEN1 (1) TO ARTIMG1.
     MOVE "ImageName" OF IMAGEN1 (2) TO ARTIMG2.
     MOVE "ImageName" OF IMAGEN1 (3) TO ARTIMG3.
     MOVE "ImageName" OF IMAGEN1 (4) TO ARTIMG4.
     MOVE "ImageName" OF IMAGEN1 (5) TO ARTIMG5.
     MOVE "ImageName" OF IMAGEN1 (6) TO ARTIMG6.

     IF "Caption" OF INFORMACION = " Nuevo Registro"  *> Es un Alta
        MOVE "N" TO ARTELI
     END-IF.

     EXEC SQL
          INSERT INTO ARTICULOS
                 VALUES (:ARTCOD,
                         :ARTDES,
                         :ARTFAM,
                         :ARTPRO,
                         :ARTIVA,
```

```
                        :ARTPRE1,
                        :ARTPRE2,
                        :ARTPRE3,
                        :ARTOBS,
                        :ARTACT,
                        :ARTIMG1,
                        :ARTIMG2,
                        :ARTIMG3,
                        :ARTIMG4,
                        :ARTIMG5,
                        :ARTIMG6,
                        :ARTFEC,
                        :ARTFEC1,
                        :ARTELI
                       )
     END-EXEC.

     IF SQLSTATE = "23000"    *> Ya existe el registro
        PERFORM REGRABAR
     END-IF.

     EXEC SQL
          COMMIT
     END-EXEC.

     INITIALIZE REG-ART.
     CALL "INHABILITAR".
     CALL "VACIAR".
     MOVE 1 TO "CurrentPage" OF TAB1.
     INVOKE CAMPO1 "SetFocus".
     EXIT PROGRAM.
*
 REGRABAR.
*
     EXEC SQL
          UPDATE ARTICULOS
                 SET ARTDES  = :ARTDES,
                     ARTFAM  = :ARTFAM,
                     ARTPRO  = :ARTPRO,
                     ARTIVA  = :ARTIVA,
                     ARTPRE1 = :ARTPRE1,
                     ARTPRE2 = :ARTPRE2,
                     ARTPRE3 = :ARTPRE3,
                     ARTOBS  = :ARTOBS,
                     ARTACT  = :ARTACT,
                     ARTIMG1 = :ARTIMG1,
                     ARTIMG2 = :ARTIMG2,
                     ARTIMG3 = :ARTIMG3,
                     ARTIMG4 = :ARTIMG4,
                     ARTIMG5 = :ARTIMG5,
                     ARTIMG6 = :ARTIMG6,
                     ARTFEC  = :ARTFEC,
                     ARTFEC1 = :ARTFEC1
                WHERE ARTCOD = :ARTCOD
     END-EXEC.
```

Como consideraciones a éste código, siempre que queráis ir al *campo1*, cómo éste pertenece a la *pestaña1*, es necesario indicarlo antes de pasarle el foco *campo1*, o nos devolverá un error, para ello usamos primeramente la instrucción `MOVE 1 TO "CurrentPage" OF TAB1` para indicarle que la pestaña "*en uso*", va a ser la 1, (en este caso). Y por otro lado, cuando hacemos una modificación en la BD, (un alta, modificación, borrado, etc.), antes de volver a hacer cualquier otra operación con la BD, hay que ejecutar el código:

```
     EXEC SQL
          COMMIT
     END-EXEC.
```

O de lo contrario, no se guardará lo que hayamos hecho en la BD, y nos volveremos locos repasando código que está bien programado, pero al que le falta una instrucción. Aunque cuando definimos el conector, marcamos la opción de *AutoCommit*, no sé la razón, pero no le hace caso. Con COMMIT, le indicamos a SQL, que actualice todos

los procesos que tiene en marcha, si lo que queremos es volver atrás una operación de SQL, cambiamos *COMMIT* por *ROLLBACK*, por supuesto, si ya hemos ejecutado un COMMIT, no hay vuelta atrás que valga.

Evento *Click* del botón BTCANCEL:

```
 ENVIRONMENT     DIVISION.
  DATA            DIVISION.
  WORKING-STORAGE SECTION.
  PROCEDURE       DIVISION.
*
      ADD POW-DMICONQUESTION POW-DMYESNO POW-DMDEFBUTTON2
          GIVING ESTILO.

      INVOKE POW-SELF "DisplayMessage"
             USING "¿Cancelar?, Perderá las modificaciones no guardadas ..."
                   WS-ATENCION
                   ESTILO
             RETURNING RETORNO
      END-INVOKE.

      IF RETORNO = POW-DMRNO
         EXIT PROGRAM
      END-IF.

      CALL "VACIAR".
      CALL "INHABILITAR".
      MOVE 1 TO "CurrentPage" OF TAB1.
      INVOKE TAB1 "SetFocus".
      INVOKE CAMPO1 "SetFocus".
      EXIT PROGRAM.
```

Evento *Click* del botón BTCONSUL:

```
 ENVIRONMENT     DIVISION.
  DATA            DIVISION.
  WORKING-STORAGE SECTION.
  01  I                        PIC 99 BINARY.
  01  WTEX                     PIC X(50).
  PROCEDURE       DIVISION.
*
      MOVE SPACES TO AHIVA15T
      INVOKE POW-SELF "CallForm" USING "F-CON-ARTICULOS"

      IF AHIVA15T = SPACES
         INVOKE BTCONSUL "SetFocus"
         EXIT PROGRAM
      END-IF.

      MOVE AHIVA15T TO "Text" OF CAMPO1.
      MOVE AHIVA15T TO ARTCOD.

      IF AHIVA15T = SPACES
         MOVE 1 TO "CurrentPage" OF TAB1
         INVOKE CAMPO1 "SetFocus"
         EXIT PROGRAM
      END-IF.

      EXEC SQL
           SELECT * FROM ARTICULOS
                    WHERE BINARY ARTCOD = :ARTCOD
                          INTO :ARTCOD,
                               :ARTDES,
                               :ARTFAM,
                               :ARTPRO,
                               :ARTIVA,
                               :ARTPRE1,
```

```
                                :ARTPRE2,
                                :ARTPRE3,
                                :ARTOBS,
                                :ARTACT,
                                :ARTIMG1,
                                :ARTIMG2,
                                :ARTIMG3,
                                :ARTIMG4,
                                :ARTIMG5,
                                :ARTIMG6,
                                :ARTFEC,
                                :ARTFEC1,
                                :ARTELI
     END-EXEC.

     IF SQLSTATE NOT = "00000"      *> Ha habido un error
        MOVE SPACES TO WTEX
        COMPUTE I = FUNCTION STORED-CHAR-LENGTH (ARTCOD)

        STRING "El artículo "
               ARTCOD (1:I)
               ", no existe ..." DELIMITED BY SIZE INTO WTEX
        END-STRING

        ADD POW-DMICONERROR POW-DMOK GIVING ESTILO
        INVOKE POW-SELF "DisplayMessage"
               USING WTEX
                     WS-ATENCION
                     ESTILO
        END-INVOKE
        CALL "INHABILITAR"
        CALL "VACIAR"
        MOVE 1 TO "CurrentPage" OF TAB1
        INVOKE CAMPO1 "SetFocus"
        EXIT PROGRAM
     ELSE
        CALL "HABILITAR"
        CALL "VACIAR"
        CALL "EXISTE"
        MOVE 1 TO "CurrentPage" OF TAB1
        INVOKE CAMPO2 "SetFocus"
        EXIT PROGRAM
     END-IF.
```

Evento *Click* del botón BTBORRAR:

```
 ENVIRONMENT     DIVISION.
  DATA            DIVISION.
  WORKING-STORAGE SECTION.
  01   I                        PIC 99 BINARY.
  01   WTEX                     PIC X(60).
  PROCEDURE       DIVISION.
*
     COMPUTE I = FUNCTION STORED-CHAR-LENGTH (ARTCOD).
     MOVE SPACES TO WTEX.
*
     IF ARTELI = "S"    *> La ficha ya está marcada como eliminado
        PERFORM BORRADO
        MOVE 1 TO "CurrentPage" OF TAB1
        CALL "VACIAR"
        CALL "INHABILITAR"
        INVOKE CAMPO1 "SetFocus"
        EXIT PROGRAM
     END-IF.

     ADD POW-DMICONQUESTION POW-DMYESNO POW-DMDEFBUTTON2
```

```
          GIVING ESTILO.

      STRING "¿Borrar el Artículo "
             ARTCOD (1:I)
             "?" DELIMITED BY SIZE INTO WTEX
      END-STRING.

      INVOKE POW-SELF "DisplayMessage"
             USING WTEX
                   WS-ATENCION
                   ESTILO
             RETURNING RETORNO
      END-INVOKE.

      IF RETORNO = POW-DMRNO
         EXIT PROGRAM
      END-IF.

      EXEC SQL
           UPDATE ARTICULOS
                  SET ARTELI = 'S'
                  WHERE ARTCOD = :ARTCOD
      END-EXEC.

      EXEC SQL
           COMMIT
      END-EXEC.

      CALL "VACIAR".
      CALL "INHABILITAR".
      MOVE 1 TO "CurrentPage" OF TAB1.
      INVOKE CAMPO1 "SetFocus".
      EXIT PROGRAM.
*
 BORRADO.
      STRING "¿Borrar DEFINITIVAMENTE el Artículo "
             ARTCOD (1:I)
             "?" DELIMITED BY SIZE INTO WTEX
      END-STRING.

      ADD POW-DMICONQUESTION POW-DMYESNO POW-DMDEFBUTTON2
          GIVING ESTILO.

      INVOKE POW-SELF "DisplayMessage"
             USING WTEX
                   WS-ATENCION
                   ESTILO
             RETURNING RETORNO
      END-INVOKE.

      IF RETORNO = POW-DMRNO
         EXIT PROGRAM
      END-IF.

      EXEC SQL
           DELETE FROM ARTICULOS
                  WHERE ARTCOD = :ARTCOD
      END-EXEC.

      EXEC SQL
           COMMIT
      END-EXEC.

      IF SQLSTATE NOT = "00000"
         ADD POW-DMICONERROR POW-DMOK GIVING ESTILO
         INVOKE POW-SELF "DisplayMessage"
                USING SQLSTATE
                      SQLMSG
                      ESTILO
```

```
        END-INVOKE
     ELSE
        MOVE SPACES TO WTEX
        STRING "El Artículo "
               ARTCOD (1:I)
               ", se ha BORRADO DEFINITIVAMENTE ..."
               DELIMITED BY SIZE INTO WTEX
        END-STRING
        ADD POW-DMICONINFORMATION POW-DMOK GIVING ESTILO
        INVOKE POW-SELF "DisplayMessage"
               USING WTEX
                     WS-ATENCION
                     ESTILO
        END-INVOKE
     END-IF.
```

Consideraciones sobre el borrado de fichas. Si es la primera vez que se elimina una ficha, ésta se marca como eliminada en el campo **ARTELI**, de la tabla en la BD, si la ficha ya está marcada, como eliminada, entonces se elimina definitivamente de la BD.

Evento *Click* del botón BTIMPRIMIR:

```
 ENVIRONMENT     DIVISION.
  DATA            DIVISION.
  WORKING-STORAGE SECTION.
  PROCEDURE       DIVISION.
*
  INICIO.
      INVOKE POW-SELF "Alarm" USING POW-MBEXCLAMATION.
          ADD POW-DMICONQUESTION POW-DMYESNO GIVING ESTILO.

          INVOKE POW-SELF "DisplayMessage"
                 USING "¿ Imprimir PANTALLA ?"
                       WS-ATENCION
                       ESTILO
                 RETURNING RETORNO
          END-INVOKE.

          IF RETORNO = POW-DMRYES
             PERFORM IMPRIMIR
          END-IF.

          EXIT PROGRAM.
  IMPRIMIR.
          INVOKE PRINT1 "PrintForm".
```

Evento *Click* del botón BTLISTAR:

```
 ENVIRONMENT     DIVISION.
  DATA            DIVISION.
  WORKING-STORAGE SECTION.
  PROCEDURE       DIVISION.
*
      INVOKE POW-SELF "CallForm" USING "F-LIS-ART".
```

Evento *Click* del botón BTSALIR:

```
 ENVIRONMENT     DIVISION.
  DATA            DIVISION.
  WORKING-STORAGE SECTION.
  PROCEDURE       DIVISION.
```

```
*
     CALL "PARA-SALIR".
```

Botones interiores del Tab1

Evento *Click* del botón BTB1:

```
 ENVIRONMENT     DIVISION.
  DATA            DIVISION.
  WORKING-STORAGE SECTION.
  01  I                        PIC 99 BINARY.
  01  WTEX                     PIC X(50).
  PROCEDURE       DIVISION.
*
     IF "Text" OF CAMPO1 = SPACES
        MOVE "Text" OF CAMPO1 TO AHIVA15T
        MOVE SPACES TO AHIVA15T
     END-IF.

     MOVE "Text" OF CAMPO1 TO AHIVA15T
     INVOKE POW-SELF "CallForm" USING "F-CON-ARTICULOS"
     MOVE AHIVA15T TO "Text" OF CAMPO1
     MOVE AHIVA15T TO ARTCOD

     IF AHIVA15T = SPACES
        INVOKE CAMPO1 "SetFocus"
        EXIT PROGRAM
     END-IF.

     EXEC SQL
          SELECT * FROM ARTICULOS
                    WHERE BINARY ARTCOD = :ARTCOD
                          INTO :ARTCOD,
                               :ARTDES,
                               :ARTFAM,
                               :ARTPRO,
                               :ARTIVA,
                               :ARTPRE1,
                               :ARTPRE2,
                               :ARTPRE3,
                               :ARTOBS,
                               :ARTACT,
                               :ARTIMG1,
                               :ARTIMG2,
                               :ARTIMG3,
                               :ARTIMG4,
                               :ARTIMG5,
                               :ARTIMG6,
                               :ARTFEC,
                               :ARTFEC1,
                               :ARTELI
     END-EXEC.

     IF SQLSTATE NOT = "00000"      *> Ha habdo un error
        MOVE SPACES TO WTEX
        COMPUTE I = FUNCTION STORED-CHAR-LENGTH (ARTCOD)

        STRING "El artículo "
               ARTCOD (1:I)
               ", no existe ..." DELIMITED BY SIZE INTO WTEX
        END-STRING

        ADD POW-DMICONERROR POW-DMOK GIVING ESTILO
        INVOKE POW-SELF "DisplayMessage"
               USING WTEX
                     WS-ATENCION
                     ESTILO
```

```
        END-INVOKE
        CALL "INHABILITAR"
        CALL "VACIAR"
        INVOKE CAMPO1 "SetFocus"
        EXIT PROGRAM
     ELSE
        CALL "HABILITAR"
        CALL "VACIAR"
        CALL "EXISTE"
        INVOKE CAMPO2 "SetFocus"
        EXIT PROGRAM
     END-IF.
```

Evento *Click* de los botones BTB1 y BTB2:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 LINKAGE         SECTION.
 01  POW-ARG-KEYASCII PIC S9(4) COMP-5.
 PROCEDURE       DIVISION USING POW-ARG-KEYASCII.
*
     EVALUATE POW-ARG-KEYASCII
              WHEN 27   *> Tecla Escape
                   CALL "PARA-SALIR"
     END-EVALUATE.
```

Evento *Click* del botón BTB2:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.

     MOVE SPACES TO AHIVA3T.

     INVOKE POW-SELF "CallForm" USING "F-CON-FAMILIAS" "M-FAMILIAS".
     MOVE AHIVA3T TO "Text" OF CAMPO3.

     IF AHIVA3T = SPACES
        MOVE SPACES TO "Caption" OF INFORMACION1
        INVOKE CAMPO3 "SetFocus"
        EXIT PROGRAM
     ELSE
        MOVE AHIVA3T TO FAMCOD
        READ FAMILIAS NO LOCK
             INVALID KEY
                     MOVE " No existe ..." TO "Caption" OF INFORMACION1
                     INVOKE CAMPO3 "SetFocus"
                     EXIT PROGRAM
             NOT INVALID KEY
                 MOVE FAMDES TO "Caption" OF INFORMACION1
        END-READ
        INVOKE CAMPO4 "SetFocus"
        EXIT PROGRAM
     END-IF.
```

Otros eventos de los Controles

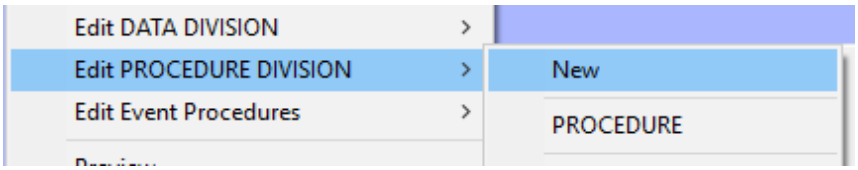
Hasta aquí, la programación de todos los eventos de todos los elementos de nuestro form, hay otros muchos eventos en los componentes, que como en nuestro caso no los vamos a utilizar, lógicamente no los programamos, aunque eso no quiere decir que no los vayamos a utilizar nunca.

- **Change**: Cuando se produce algún cambio en el elemento. (El que sea, da lo mismo).
- **DblClick**: Cuando hacemos doble-click en el elemento. Tener en cuenta que, si tenemos algo programado en el evento *Click*, se ejecutará antes que el evento *DblClick*.
- **Edit**: Cuando editamos el elemento, (normalmente un campo del tipo *Edit*).
- **GotFocus**: Cuando el control adquiere el foco.
- **KeyDown**: Cuando pulsamos una tecla.
- **KeyUp**: Cuando “soltamos” una tecla pulsada.
- **LostFocus**: Cuando un control pierde el foco.
- **MouseDown**: Cuando hacemos click en un control.
- **MouseMove**: Cuando movemos el puntero del ratón por encima de un control
- **MouseUp**: Cuando “soltamos” el botón con el que hemos hecho click en un control.

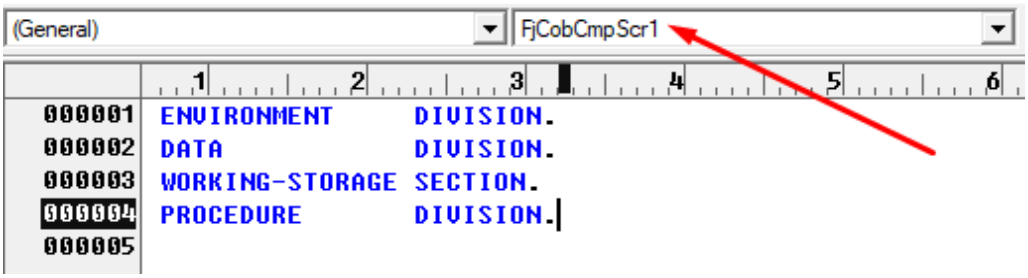
Éstos son los más usuales, aunque hay más que veremos más adelante con otros controles.

Subrutinas

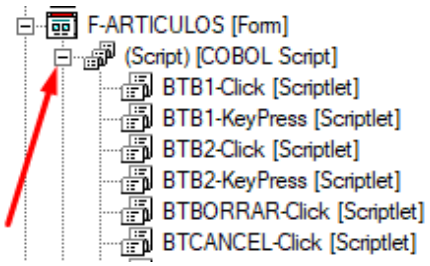
Sólo nos quedan las subrutinas, que en éste caso se llaman ***Procedures***, la sintáxis de llamada es la misma que en **COBOL ANSI**, `CALL "subrutina" [USING Variable_1, variable_2 … variable_n]`, si utilizamos la cláusula **USING**, las variables que le pasemos, han de estar definidas en la **LINKAGE SECTION** del programa llamado, como he comentado antes, igual que en **COBOL** de toda la vida.
Para crear una nueva subrutina, hacemos click con el botón derecho en cualquier parte del form que no contenga ningún control o elemento, y seleccionamos *Edit PROCEDURE DIVISION → New*.



Una vez realizado ésto, se nos abrirá una nueva ventana, en la que introduciremos el código fuente de la nueva subrutina, en la imagen siguiente, con la flecha roja os indico el nombre que pone por defecto PWC a las subrutinas, después tendremos que cambiarlo por el que hayamos utilizado en nuestro **CALL**.

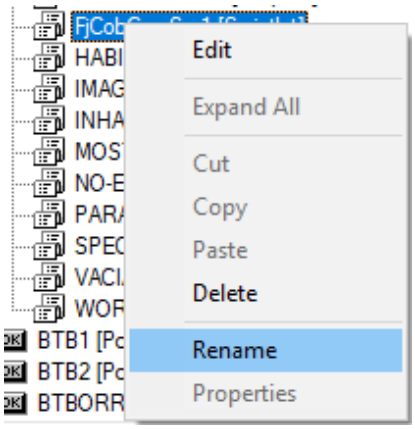


Para ello, vamos al árbol de la izquierda, buscamos la rama *(Script) [COBOL Script]*, la desplegamos,



Desplegamos el árbol de la izquierda.

y buscamos el elemento *FjCobCmpScr1*, (van por orden alfabético), hacemos click con el botón derecho, seleccionamos *Rename* en introducimos el nombre deseado.



Así que vamos con todas las subrutinas que necesitamos.

Subrutina "EXISTE"

```
 ENVIRONMENT     DIVISION.
  DATA            DIVISION.
  WORKING-STORAGE SECTION.
  01  I                        PIC 99 BINARY.
  01  WTEX                     PIC X(90).
  PROCEDURE       DIVISION.
*
      MOVE SPACES TO WTEX.
      COMPUTE I = FUNCTION STORED-CHAR-LENGTH (ARTCOD).

      STRING "Fichero de Artículos: "
             ARTCOD (1:I)
             " - "
             ARTDES DELIMITED BY SIZE INTO WTEX
      END-STRING.

      MOVE POW-COLOR-DARKRED    TO "ForeColor" OF INFORMACION.
      MOVE WTEX                 TO "Caption"   OF POW-SELF.
      MOVE " Modifica Registro" TO "Caption"   OF INFORMACION.

      IF ARTELI = "S"    *> El Registro está marcado como ELIMINADO
         MOVE 1 TO "Visible" OF IMGELI
      ELSE
         MOVE 0 TO "Visible" OF IMGELI
      END-IF.
*
      MOVE ARTDES  TO "Text" OF CAMPO2.
      MOVE ARTFAM  TO "Text" OF CAMPO3.
      CALL "MOSTRAR-FAM".
      MOVE ARTPRO  TO "Text" OF CAMPO4.
      MOVE ARTIVA  TO "Text" OF CAMPO5.
      MOVE ARTPRE1 TO "Text" OF CAMPO6.
      MOVE ARTPRE2 TO "Text" OF CAMPO7.
      MOVE ARTPRE3 TO "Text" OF CAMPO8.

      MOVE ARTFEC TO FECHA.
      MOVE CORR FECHA TO FECHA1.
      MOVE FECHA1 TO "Text" OF CAMPO9.

      MOVE ARTFEC1 TO FECHA.
      MOVE CORR FECHA TO FECHA1.
      MOVE FECHA1 TO "Text" OF CAMPO10.

      MOVE ARTOBS TO "Text" OF CAMPO11.

      IF ARTACT = "S"
         MOVE POW-TRISTATE-CHECKED   TO "Value" OF CHECK1
      ELSE
         MOVE POW-TRISTATE-UNCHECKED TO "Value" OF CHECK1
      END-IF.
*
      IF ARTIMG1 NOT = SPACES
         MOVE ARTIMG1 TO "ImageName"   OF IMAGEN1 (1)
         MOVE ARTIMG1 TO "ToolTipText" OF IMAGEN1 (1)
      END-IF.
      INVOKE IMAGEN1 (1) "Refresh".

      IF ARTIMG2 NOT = SPACES
         MOVE ARTIMG2 TO "ImageName"   OF IMAGEN1 (2)
         MOVE ARTIMG2 TO "ToolTipText" OF IMAGEN1 (2)
      END-IF.
      INVOKE IMAGEN1 (2) "Refresh".

      IF ARTIMG3 NOT = SPACES
         MOVE ARTIMG3 TO "ImageName"   OF IMAGEN1 (3)
```

```
        MOVE ARTIMG3 TO "ToolTipText" OF IMAGEN1 (3)
     END-IF.
     INVOKE IMAGEN1 (3) "Refresh".

     IF ARTIMG4 NOT = SPACES
        MOVE ARTIMG4 TO "ImageName"   OF IMAGEN1 (4)
        MOVE ARTIMG4 TO "ToolTipText" OF IMAGEN1 (4)
     END-IF.
     INVOKE IMAGEN1 (4) "Refresh".

     IF ARTIMG5 NOT = SPACES
        MOVE ARTIMG5 TO "ImageName"   OF IMAGEN1 (5)
        MOVE ARTIMG5 TO "ToolTipText" OF IMAGEN1 (5)
     END-IF.
     INVOKE IMAGEN1 (5) "Refresh".

     IF ARTIMG6 NOT = SPACES
        MOVE ARTIMG6 TO "ImageName"   OF IMAGEN1 (6)
        MOVE ARTIMG6 TO "ToolTipText" OF IMAGEN1 (6)
     END-IF.
     INVOKE IMAGEN1 (6) "Refresh".
```

Subrutina "HABILITAR"

```
 ENVIRONMENT     DIVISION.
  DATA            DIVISION.
  WORKING-STORAGE SECTION.
  01   I                        PIC 9 BINARY.
  PROCEDURE       DIVISION.
*
     MOVE 1 TO "Enabled"  OF CAMPO2.
     MOVE 1 TO "Enabled"  OF CAMPO3.
     MOVE 1 TO "Enabled"  OF CAMPO4.
     MOVE 1 TO "Enabled"  OF CAMPO5.
     MOVE 1 TO "Enabled"  OF CAMPO6.
     MOVE 1 TO "Enabled"  OF CAMPO7.
     MOVE 1 TO "Enabled"  OF CAMPO8.
     MOVE 1 TO "Enabled"  OF CAMPO9.
     MOVE 1 TO "Enabled"  OF CAMPO10.
     MOVE 1 TO "Enabled"  OF CAMPO11.
     MOVE 1 TO "Enabled"  OF CHECK1.
     MOVE 1 TO "Enabled"  OF BTB2.
*
     PERFORM VARYING I FROM 1 BY 1 UNTIL I > 6
             MOVE 1 TO "Enabled" OF IMAGEN1 (I)
             MOVE 1 TO "Enabled" OF BTIMG1  (I)
             MOVE 1 TO "Enabled" OF BTIMG2  (I)
     END-PERFORM.
*
     MOVE 1 TO "Enabled" OF BTGRABAR.
     MOVE 1 TO "Enabled" OF BTCANCEL.
     MOVE 1 TO "Enabled" OF BTBORRAR.
     MOVE 1 TO "Enabled" OF BTIMPRIMIR.
```

Subrutina "INHABILITAR"

```
 ENVIRONMENT     DIVISION.
  DATA            DIVISION.
  WORKING-STORAGE SECTION.
  01   I                        PIC 9 BINARY.
  PROCEDURE       DIVISION.
*
     MOVE 0 TO "Enabled"  OF CAMPO2.
```

```
     MOVE 0 TO "Enabled"  OF CAMPO3.
     MOVE 0 TO "Enabled"  OF CAMPO4.
     MOVE 0 TO "Enabled"  OF CAMPO5.
     MOVE 0 TO "Enabled"  OF CAMPO6.
     MOVE 0 TO "Enabled"  OF CAMPO7.
     MOVE 0 TO "Enabled"  OF CAMPO8.
     MOVE 0 TO "Enabled"  OF CAMPO9.
     MOVE 0 TO "Enabled"  OF CAMPO10.
     MOVE 0 TO "Enabled"  OF CAMPO11.
     MOVE 0 TO "Enabled"  OF CHECK1.
     MOVE 0 TO "Enabled"  OF BTB2.
*
     PERFORM VARYING I FROM 1 BY 1 UNTIL I > 6
             MOVE 0 TO "Enabled" OF IMAGEN1 (I)
             MOVE 0 TO "Enabled" OF BTIMG1  (I)
             MOVE 0 TO "Enabled" OF BTIMG2  (I)
     END-PERFORM.
*
     MOVE 0 TO "Enabled" OF BTGRABAR.
     MOVE 0 TO "Enabled" OF BTCANCEL.
     MOVE 0 TO "Enabled" OF BTBORRAR.
     MOVE 0 TO "Enabled" OF BTIMPRIMIR.
```

Subrutina "MOSTRAR-FAM"

```
ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.

     MOVE "Text" OF CAMPO3 TO FAMCOD.
     READ FAMILIAS NO LOCK KEY IS FAMCOD
          INVALID KEY
                  MOVE "No Existe ..." TO "Caption" OF INFORMACION1
          NOT INVALID KEY
                  MOVE FAMDES TO "Caption" OF INFORMACION1
     END-READ.
```

Subrutina "NO-EXISTE"

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  I                        PIC 99 BINARY.
 01  WTEX                     PIC X(90).
 PROCEDURE       DIVISION.
*
     CALL "VACIAR".
     MOVE SPACES TO WTEX.
     COMPUTE I = FUNCTION STORED-CHAR-LENGTH (ARTCOD).

     MOVE POW-COLOR-BLACK TO "ForeColor" OF INFORMACION.
     STRING "Fichero de Artículos: "
            ARTCOD (1:I)
            " - Nuevo Registro" DELIMITED BY SIZE INTO WTEX
     END-STRING.

     MOVE WTEX              TO "Caption" OF POW-SELF.
     MOVE " Nuevo Registro" TO "Caption" OF INFORMACION.
```

Subrutina "PARA-SALIR"

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.
*
     ADD POW-DMICONQUESTION POW-DMYESNO GIVING ESTILO.
     INVOKE POW-SELF "DisplayMessage"
            USING "¿Salir del programa?"
                  WS-ATENCION
                  ESTILO
           RETURNING RETORNO
     END-INVOKE.

     IF RETORNO = POW-DMRNO
        EXIT PROGRAM
     END-IF.

     INVOKE POW-SELF "CloseForm".
```

Subrutina "VACIAR"

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  I                          PIC 9 BINARY.
 PROCEDURE       DIVISION.
*
     MOVE SPACES                TO "Text"  OF CAMPO2.
     MOVE SPACES                TO "Text"  OF CAMPO3.
     MOVE 0                     TO "Text"  OF CAMPO4.
     MOVE 0                     TO "Text"  OF CAMPO5.
     MOVE 0                     TO "Text"  OF CAMPO6.
     MOVE 0                     TO "Text"  OF CAMPO7.
     MOVE 0                     TO "Text"  OF CAMPO8.
     MOVE 0                     TO "Text"  OF CAMPO10.
     MOVE SPACES                TO "Text"  OF CAMPO11.
     MOVE POW-TRISTATE-CHECKED TO "Value" OF CHECK1.
*
     PERFORM VARYING I FROM 1 BY 1 UNTIL I > 6
             MOVE SPACES TO "ImageName"   OF IMAGEN1 (I)
             MOVE SPACES TO "ToolTipText" OF IMAGEN1 (I)
             INVOKE IMAGEN1 (I) "Refresh"
     END-PERFORM.
*
     MOVE SPACES                  TO "Caption" OF INFORMACION.
     MOVE SPACES                  TO "Caption" OF INFORMACION1.
     MOVE "Fichero de Artículos" TO "Caption" OF POW-SELF.
     MOVE 0                       TO "Visible" OF IMGELI
```

Variables propias de PWC.

PWC tiene sus propias variables internas, que podemos utilizarlas tanto por su nombre, (por ejemplo POW-TRUE, que vale 1, POW-OFF que vale 0, etc.), como por su valor, por lo que es lo mismo, escribir **`MOVE 1 TO "Visible" OF BTB1`** que **`MOVE POW-TRUE TO "Visible" OF BTB1`** o **`MOVE 0 TO "Enabled" OF CAMPO2`** que **`MOVE POW-FALSE TO "Enabled" OF CAMPO2`**.

Éste es el listado de todas la variables con sus respectivos valores, están sacadas directamente de la ayuda del PWC, así que no creo que haya ningún error: Ánimo y paciencia, que son muchas.

Align **Property**

Value	Constant
0 - None	POW-ALIGNNONE
1 - Top	POW-ALIGNTOP
2 - Bottom	POW-ALIGNBOTTOM
3 - Left	POW-ALIGNLEFT
4 - Right	POW-ALIGNRIGHT

Alignment Property (CheckBox, OptionButton)

Value	Constant
0 - Left	POW-CAPTIONALIGNMENT-LEFT
1 - Right	POW-CAPTIONALIGNMENT-RIGHT

Alignment Property (Column, TextBox)

Value	Constant
0 - Left	POW-TEXTALIGNMENT-LEFT
1 - Center	POW-TEXTALIGNMENT-CENTER
2 - Right	POW-TEXTALIGNMENT-RIGHT

Alignment Property (Frame)

Value	Constant
99 – Auto	POW-ALIGNMENT-AUTO

Alignment Property (Frame, StaticText, TableColumn)

Value	Constant
0 - Top/Left	POW-ALIGNMENT-TOPLEFT
1 - Top/HCenter	POW-ALIGNMENT-TOPHCENTER
2 - Top/Right	POW-ALIGNMENT-TOPRIGHT
4 - VCenter/Left	POW-ALIGNMENT-VCENTERLEFT

5 - VCenter/HCenter	POW-ALIGNMENT-VCENTERHCENTER
6 - VCenter/Right	POW-ALIGNMENT-VCENTERRIGHT
8 - Bottom/Left	POW-ALIGNMENT-BOTTOMLEFT
9 - Bottom/HCenter	POW-ALIGNMENT-BOTTOMHCENTER
10 - Bottom/Right	POW-ALIGNMENT-BOTTOMRIGHT

Appearance Property

Value	Constant
0 - Flat	POW-APPEARANCE-FLAT
1 - 3D	POW-APPEARANCE-3D

Arrange Property

Value	Constant
0 - None	POW-LVICON-AUTO
1 - Left	POW-LVICON-LEFT
2 - Top	POW-LVICON-TOP

BackStyle Property (Shape)

Value	Constant
2 - Half transparent	POW-BACKSTYLE-TRANSLUCENT

BackStyle Property (Shape, TextBox, Others)

Value	Constant
0 - Transparent	POW-BACKSTYLE-TRANSPARENT
1 - Opaque	POW-BACKSTYLE-OPAQUE

BorderStyle Property (Form)

Value	Constant
0 - Variable	POW-BORDER-VARIABLE
1 - Fixed	POW-BORDER-FIXED
2 - Dialog Frame	POW-BORDER-DIALOG
3 - Variable Tool	POW-BORDER-VARIABLETOOL
4 - Fixed Tool	POW-BORDER-FIXEDTOOL
5 - No Frame	POW-BORDER-NOFRAME

BorderStyle Property (Frame/Shape/Others)

Value	Constant
0 - None	POW-BORDER-NONE
1 - Solid	POW-BORDER-SOLID

BorderStyle Property (Shape)

Value	Constant
2 - Dash	POW-BORDER-DASH
3 - Dot	POW-BORDER-DOT
4 - Dash dot	POW-BORDER-DASHDOT
5 - Dash double dot	POW-BORDER-DASHDOTDOT
6 - InsideFrame	POW-BORDER-INSIDEFRAME

ColorMap Property

Value	Constant
0 - No	POW-COLORMAP-NO
1 - Back	POW-COLORMAP-BACK
2 - All	POW-COLORMAP-ALL

CommandButtonFunctionKeyMode Property

Value	Constant
0 : Normal	POW-FUNCTIONKEYMODE-NORMAL
1 : Set focus to CommandButton control	POW-FUNCTIONKEYMODE-FOCUS

CommandType Property

Value	Constant
1 - adCmdText	POW-ADODB-ADCMDTEXT
2 - adCmdTable	POW-ADODB-ADCMDTABLE
4 - adCmdStoredProc	POW-ADODB-ADCMDSTOREDPROC
8 - adCmdUnknown	POW-ADODB-ADCMDUNKNOWN

CommitMode Property

Value	Constant
0 - Automatic	POW-DBCOMMIT-AUTO

1 - Manual	POW-DBCOMMIT-MANUAL

ConcurrencyCursor Property

Value	Constant
1 - SQL_CONCUR_READ_ONLY	POW-DBCONCUR-READONLY
2 - SQL_CONCUR_LOCK	POW-DBCONCUR-LOCK
3 - SQL_CONCUR_OPT_ROWVER	POW-DBCONCUR-OPTROWVER
4 - SQL_CONCUR_OPT_VALUES	POW-DBCONCUR-OPTVALUES

ConnectMode Property

Value	Constant
0 - adModeUnknown	POW-ADODB-ADMODEUNKNOWN
1 - adModeRead	POW-ADODB-ADMODEREAD
2 - adModeWrite	POW-ADODB-ADMODEWRITE
3 - adModeReadWrite	POW-ADODB-ADMODEREADWRITE
4 - adModeShareDenyRead	POW-ADODB-ADMODESHAREDENYREAD
8 - adModeShareDenyWrite	POW-ADODB-ADMODESHAREDENYWRITE
12 - adModeShareExclusive	POW-ADODB-ADMODESHAREEXCLUSIVE
16 - adModeShareDenyNone	POW-ADODB-ADMODESHAREDENYNONE

CursorLocation Property

Value	Constant
2 - adUserServer	POW-ADODB-ADUSERSERVER
3 - adUserClient	POW-ADODB-ADUSERCLIENT

CursorType Property (ADODataSource)

Value	Constant
1 - adOpenKeyset	POW-ADODB-ADOPENKEYSET

2 - adOpenDynamic	POW-ADODB-ADOPENDYNAMIC
3 - adOpenStatic	POW-ADODB-ADOPENSTATIC

CursorType Property (DBAccess)

Value	Constant
0 - SQL_CURSOR_FORWARD_ONLY	POW-DBCURSOR-FORWARDONLY
1 - SQL_CURSOR_KEYSET_DRIVEN	POW-DBCURSOR-KEYSETDRIVEN
2 - SQL_CURSOR_DYNAMIC	POW-DBCURSOR-DYNAMIC
3 - SQL_CURSOR_STATIC	POW-DBCURSOR-STATIC

DateStyle Property

Value	Constant
0 - Month dd yyyy	POW-DATESTYLE-MONTHDDYYYY
1 - yyyy/ MM /dd	POW-DATESTYLE-YYYYMMDD-SLASH
2 - yyyy. MM.dd	POW-DATESTYLE-YYYYMMDD-PERIOD
3 - MM /dd/yyyy	POW-DATESTYLE-MMDDYYYY-SLASH
4 - MM.dd.yyyy	POW-DATESTYLE-MMDDYYYY-PERIOD
5 - Mon dd yyyy	POW-DATESTYLE-MONDDYYYY
99 - Custom	POW-DATESTYLE-CUSTOM

DDEDataStyle Property

Value	Constant
10 - Text	POW-DDETEXT
11 - Binary	POW-DDEBINARY

DDELinkStyle Property

Value	Constant
1 - Hot Link	POW-DDEHOTLINK
2 - Worm Link	POW-DDEWARMLINK

DisabledFocusAction Property

Value	Constant
0 - Normal	POW-DFA-NORMAL

1 - Ignore option button value	POW-DFA-IGNOREOPTIONVALUE

DisabledImageType Property

Value	Constant
0 - None	POW-DISABLEDIMAGE-NONE
1 - Simple	POW-DISABLEDIMAGE-SIMPLE
2 - 3D	POW-DISABLEDIMAGE-3D

EdgeStyle Property

Value	Constant
0 - Bump	POW-EDGEBUMP
1 - Etched	POW-EDGEETCHED
2 - Raised	POW-EDGERAISED
3 - Sunken	POW-EDGESUNKEN
4 - Raised Light	POW-EDGERAISEDLIGHT
5 - Sunken Light	POW-EDGESUNKENLIGHT

FileType Property

Value	Constant
0 - Standard Text File	POW-FILETYPE-NORECORD
1 - CSV Format	POW-FILETYPE-CSV
2 - Fixed Length Format	POW-FILETYPE-FIXEDLENGTH

FillStyle Property

Value	Constant
0 - None	POW-FILLSTYLE-NONE
1 - Solid	POW-FILLSTYLE-SOLID
2 - Horizontal	POW-FILLSTYLE-HORIZONTAL
3 - Vertical	POW-FILLSTYLE-VERTICAL
4 - Backward diagonal	POW-FILLSTYLE-BDIAGONAL
5 - Forward diagonal	POW-FILLSTYLE-FDIAGONAL
6 - Cross	POW-FILLSTYLE-CROSS

7 - Diagonal cross POW-FILLSTYLE-DIAGONAL

FrameStyle Property

Value	Constant
0 - GroupBox	POW-FRAME-GROUPBOX
1 - Panel	POW-FRAME-PANEL

GraphStyle Property

Value	Constant
0 - Vertical Bar	POW-GRAPHSTYLE-VERTICALBAR
1 - Horizontal Bar	POW-GRAPHSTYLE-HORIZONTALBAR
2 - Line	POW-GRAPHSTYLE-LINE
3 - Pie	POW-GRAPHSTYLE-PIE

Icon Property

Value	Constant
0 - Default	POW-ICON-DEFAULT
1 - Application	POW-ICON-APPLICATION
2 - Hand	POW-ICON-HAND
3 - Question	POW-ICON-QUESTION
4 - Exclamation	POW-ICON-EXCLAMATION
5 - Asterisk	POW-ICON-ASTERISK
6 - Winlogo	POW-ICON-WINLOGO
99 - Custom	POW-ICON-CUSTOM

ImageMode Property

Value	Constant
0 - Standard	POW-IMAGEMODE-STANDARD
1 - Stretch	POW-IMAGEMODE-STRETCH
2 - Auto	POW-IMAGEMODE-AUTO
3 - Adjust	POW-IMAGEMODE-ADJUST

IMEMode Property

Value	Constant
0 - None	POW-IMENOOPERATION

1 - On	POW-IMEON
2 - Off	POW-IMEOFF
3 - Disable	POW-IMEDISABLE
4 - Hiragana Full	POW-IMEDBCSHIRAGANA
5 - Katakana Full	POW-IMEDBCSKATAKANA
6 - Katakana Half	POW-IMESBCSKATAKANA
7 - Alphabet Full	POW-IMEDBCSALPHABET
8 - Alphabet Half	POW-IMESBCSALPHABET

Layout Property

Value	Constant
0 - Horizontal	POW-LAYOUT-HORIZONTAL
1 - Vertical	POW-LAYOUT-VERTICAL

LockType Property

Value	Constant
-1 - adLockUnspecified	POW-ADODB-ADLOCKSPECIFIED
1 - adLockReadOnly	POW-ADODB-ADLOCKREADONLY
2 - adLockPessimistic	POW-ADODB-ADLOCKPESSIMISTIC
3 - adLockOptimistic	POW-ADODB-ADLOCKOPTIMISTIC
4 - adLockBatchOptimistic	POW-ADODB-ADLOCKBATCHOPTIMISTIC

LVStyle Property

Value	Constant
0 - Large Icon	POW-LVSTYLE-LARGEICON
1 - Small Icon	POW-LVSTYLE-SMALLICON
2 - List	POW-LVSTYLE-LIST
3 - Report	POW-LVSTYLE-REPORT

MenuBreak Property

Value	Constant
0 - Not MenuBreak	POW-NOBREAK
1 - MenuBreak	POW-BREAK

2 - MenuBarBreak	POW-BARBREAK

MouseButton Parameter (MouseDown, MouseMove, MouseUp events)

Value	Constant
1 - Left button	POW-MOUSE-LBUTTON
2 - Right button	POW-MOUSE-RBUTTON
3 - Middle button	POW-MOUSE-MBUTTON

[Mouse Left button]	POW-KEY-LBUTTON	Same as POW-MOUSE-LBUTTON
[Mouse Right button]	POW-KEY-RBUTTON	Same as POW-MOUSE-RBUTTON
[Mouse Middle button]	POW-KEY-MBUTTON	Same as POW-MOUSE-MBUTTON

MousePointer Property

Value	Constant
0 - Default	POW-MP-DEFAULT
1 - Arrow	POW-MP-ARROW
2 - Cross	POW-MP-CROSS
3 - Ibeam	POW-MP-IBEAM
4 - Icon	POW-MP-ICON
5 - Size	POW-MP-SIZE
6 - SizeNESW	POW-MP-SIZENESW
7 - SizeNS	POW-MP-SIZENS
8 - SizeNWSE	POW-MP-SIZENWSE
9 - SizeWE	POW-MP-SIZEWE
10 - UpArrow	POW-MP-UPARROW
11 - Wait	POW-MP-WAIT
12 - AppStarting	POW-MP-APPSTARTING
13 - No	POW-MP-NO
14 - SizeAll	POW-MP-SIZEALL

15 - Help POW-MP-HELP

99 - Custom POW-MP-CUSTOM

MultiSelect Property (FileList/ListBox)

Value	Constant
0 - Single select	POW-MULTISELECT-NONE
1 - Multiple select	POW-MULTISELECT-SIMPLE
2 - Extended select	POW-MULTISELECT-EXTENDED

OptionButtonClickMode Property

Value	Constant
0 : Normal	POW-CLICKMODE-NORMAL
1: When left button is pressed	POW-CLICKMODE-LBUTTONDOWN

Orientation Property (Others)

Value	Constant
0 - Horizontal	POW-ORIENTATION-HORIZONTAL
1 - Vertical	POW-ORIENTATION-VERTICAL

Orientation Property (Tab)

Value	Constant
0 - Top	POW-TAB-ORIENTATION-TOP
1 - Bottom	POW-TAB-ORIENTATION-BOTTOM
2 - Left	POW-TAB-ORIENTATION-LEFT
3 - Right	POW-TAB-ORIENTATION-RIGHT

PageNoFormat Property

Value	Constant
0 - None	POW-NOPAGENO
1 - Page	POW-CURRENTPAGE
2 - Page/Total	POW-CURRENTTOT

PaperOrientation Property

Value	Constant
0 - Auto	POW-DEFAULTORIENTATION
1 - Portrait	POW-PORTRAIT

2 - Landscape	POW-LANDSCAPE

Constants_Top PaperScaleUnit Property

Value	Constant
0 - mm	POW-MM
1 - Inch	POW-INCH

PaperType Property

Value	Constant
0 - Auto	POW-DEFAULTSIZE
1 - Letter 216x279 mm/Letter 8.5x11 inch	POW-PAPERLETTER
5 - Legal 216x356 mm/Legal 8.5x14 inch	POW-PAPERLEGAL
8 - A3 297x420 mm/A3 11.5x16.5 inch	POW-PAPERA3
9 - A4 210x297 mm/A4 8.2x11.5 inch	POW-PAPERA4
11 - A5 148x210 mm/5.8x8.2 inch	POW-PAPERA5
12 - B4 257x364 mm/10x14.3 inch	POW-PAPERB4
13 - B5 182x257 mm/7x10 inch	POW-PAPERB5

PcdColorType Property

Value	Constant
0 - 24bit color	POW-PCDCOLORTYPE-24BITCOLOR
1 - 256 color	POW-PCDCOLORTYPE-256COLOR
2 - 16 color	POW-PCDCOLORTYPE-16COLOR
3 - 256 gray scale color	POW-PCDCOLORTYPE-256GRAY

PcdResolution Property

Value	Constant
0 - BASE_OVER_64 (64 x 96)	POW-PCDRESOLUTION-BASE-OVER-64
1 - BASE_OVER_16 (128 x 192)	POW-PCDRESOLUTION-BASE-OVER-16
2 - BASE_OVER_4 (256 x	POW-PCDRESOLUTION-BASE-

384)	OVER-4
3 - BASE (512 x 768)	POW-PCDRESOLUTION-BASE
4 - 4BASE (1024 x 1526)	POW-PCDRESOLUTION-4BASE
5 - 16BASE (2048 x 3072)	POW-PCDRESOLUTION-16BASE

RenderStyle Property

Value	Constant
0 - Standard	POW-RENDERSTYLE-STANDARD
1 - COBOL PICTURE	POW-RENDERSTYLE-COBOLPICTURE
2 - Date	POW-RENDERSTYLE-DATE

ScaleMode Property
ScaleModeIn/ScaleModeOut Parameters (ConvertScale method), ScaleMode Parameters (GetScreenSize/GetWorkArea methods)

Value	Constant
0 - Pixels	POW-SCALEPIXELS
1 - 1/100mm	POW-SCALEHIMETRIC
2 - 1/1000inch	POW-SCALEHIENGLISH
3 - 1/20Points	POW-SCALETWIPS

ScalingStyle Property

Value	Constant
0 - None	POW-SCALING-NONE
1 - Resize only	POW-SCALING-SIZE
2 - Reposition only	POW-SCALING-POSITION
3 - Resize and reposition	POW-SCALING-SIZEPOSITION
5 - Resize with resize font	POW-SCALING-SIZEFONT
7 - Resize and reposition with resize font	POW-SCALING-SIZEPOSITIONFONT

ShapeStyle Property

Value	Constant
0 - Rectangle	POW-SHAPESTYLE-RECTANGLE
1 - Square	POW-SHAPESTYLE-SQUARE

2 - Circle POW-SHAPESTYLE-CIRCLE

3 - Ellipse POW-SHAPESTYLE-ELLIPSE

ShiftState Parameter (KeyDown, KeyUp, PreKeyDown, PreKeyUp, MouseDown, MouseMove, MouseUp events)

Value	Constant
1 - Shift	POW-SHIFTSTATE-SHIFT
2 - Ctrl	POW-SHIFTSTATE-CTRL
4 - Alt	POW-SHIFTSTATE-ALT

SortKind Property

Value	Constant
0 - Text	POW-SORTKIND-TEXT
1 - Numeric	POW-SORTKIND-NUMERIC

SortOrder Property

Value	Constant
0 - None	POW-SORTORDER-NONE
1 - Ascending	POW-SORTORDER-ASC
2 - Descending	POW-SORTORDER-DESC

Source Property

Value	Constant
0 - Default	POW-PRINTSOURCE-DEFAULT

StartUpPosition Property

Value	Constant
0 - Auto	POW-STARTUPPOS-AUTO
1 - Center Owner	POW-STARTUPPOS-CENTEROWN
2 - Center Screen	POW-STARTUPPOS-CENTERSCR
3 - Windows Default	POW-STARTUPPOS-DEFAULT
4 - Manual	POW-STARTUPPOS-MANUAL

Style Property (Button)

Value	Constant
0 - Normal	POW-TBAR-BTN-STYLE-NORMAL
1 - Separator	POW-TBAR-BTN-STYLE-SEPARATOR

2 - CheckButton POW-TBAR-BTN-STYLE-CHECK

3 - CheckButtonGroup POW-TBAR-BTN-STYLE-CHECKGROUP

4 - DropDownButton POW-TBAR-BTN-STYLE-DROPDOWN

Style Property (ComboBox)

Value	Constant
0 - Simple	POW-COMBO-STYLE-SIMPLE
1 - DropDown	POW-COMBO-STYLE-DROPDOWN
2 - DropDownList	POW-COMBO-STYLE-DROPDOWNLIST

Style Property (Tab)

Value	Constant
0 - Tabs	POW-TAB-STYLE-TABS
1 - Buttons	POW-TAB-STYLE-BUTTONS

Style Property (Toolbar)

Value	Constant
0 - Normal	POW-TBAR-STYLE-NORMAL
1 - Flat	POW-TBAR-STYLE-FLAT
2 - FlatList	POW-TBAR-STYLE-FLATLIST

TabWidthStyle Property

Value	Constant
0 - Normal	POW-TAB-STYLE-NORMAL
1 - Fixed	POW-TAB-STYLE-FIXED

TargetType Property

Value	Constant
0 - Table/View	POW-TARGETYPE-TABLE
1 - Procedure	POW-TARGETYPE-PROCEDURE

TextCase Property

Value	Constant
0 - Normal	POW-TEXTCASE-NORMAL
1 - UpperCase	POW-TEXTCASE-UPPERCASE

2 - LowerCase	POW-TEXTCASE-LOWERCASE

TickStyle Property

Value	Constant
0 - No Ticks	POW-TICKSTYLE-NOTICKS
1 - Bottom/Right	POW-TICKSTYLE-BOTTOMRIGHT
2 - Top/Left	POW-TICKSTYLE-TOPLEFT
3 - Both	POW-TICKSTYLE-BOTH

TimeFormat Property

Value	Constant
0 - Millisecond	POW-MCIFORMATMILLISECONDS
2 - Minute/second/frame	POW-MCIFORMATMSF
3 - Frame	POW-MCIFORMATFRAMES
10 - Track/minute/second/frame	POW-MCIFORMATTMSF

TVLineStyle Property

Value	Constant
0 - TreeLines	POW-TVLINESTYLE-TREELINES
1 - RootLines	POW-TVLINESTYLE-ROOTLINES

TVStyle Property

Value	Constant
0 - Text Only	POW-TVSTYLE-TEXTONLY
1 - Image and Text	POW-TVSTYLE-IMAGE
2 - Plus/Minus and Text	POW-TVSTYLE-PLUSMINUS
3 - Plus/Minus, Image and Text	POW-TVSTYLE-PLUSMINUSIMAGE
4 - TreeLines and Text	POW-TVSTYLE-LINE
5 - TreeLines, Image and Text	POW-TVSTYLE-LINEIMAGE
6 - TreeLines, Plus/Minus and Text	POW-TVSTYLE-LINEPLUSMINUS
7 - TreeLines, Plus/Minus, Image and Text	POW-TVSTYLE-LINEPLUSMINUSIMAGE

Value Property (CheckBox)

Value	Constant
0 - Unchecked	POW-TRISTATE-UNCHECKED
1 - Checked	POW-TRISTATE-CHECKED
2 - Gray	POW-TRISTATE-GRAY

Weight Property

Value	Constant
Normal	POW-FONTWEIGHT-NORMAL
Bold	POW-FONTWEIGHT-BOLD

WindowAdjustment Property

Value	Constant
0 - Standard	POW-MCIWINADJUST-STANDARD
1 - Adjust Control Window	POW-MCIWINADJUST-CONTROLWINDOW
2 - Adjust Device Frame	POW-MCIWINADJUST-DEVICEFRAME

WindowState Property

Value	Constant
0 - Normal	POW-WINDOWSTATE-NORMAL
1 - Maximize	POW-WINDOWSTATE-MAXIMIZE
2 - Minimize	POW-WINDOWSTATE-MINIMIZE

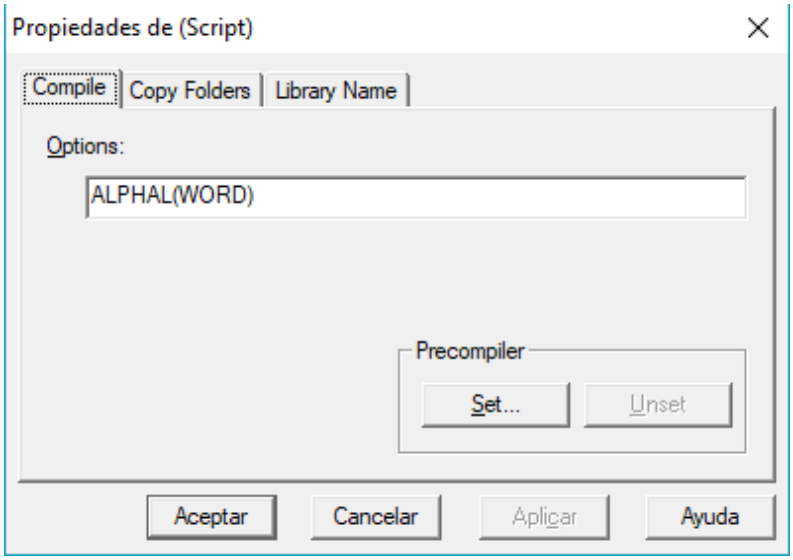
Value	Constant
0 - Puts the control at the top of the order.	POW-ZORDER-FRONT
1 - Puts the control at the bottom of the order.	

PWC y la API de Windows. ALPHAL(WORD).

Desde la versión 5, PWC soporta perfectamente llamadas a las ***APIs*** de Windows, pero para poder usarlas, tenemos que indicarlo antes mediante la instrucción **ALPHAL(WORD)**, para ello hay dos maneras, primera en el subprograma desde el que vayamos a hacer la llamada a la API antes de **IDENTIFICATION DIVISION**, lo declaramos de ésta manera:

```
@OPTIONS ALPHAL(WORD)
IDENTIFICATION    DIVISION.
```

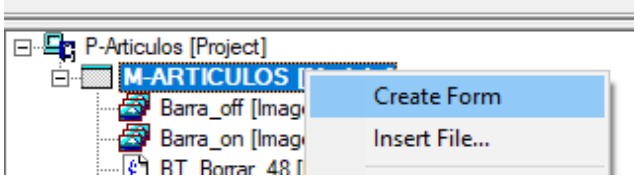
O nos dirigimos al árbol de la izquierda, hacemos click con en botón derecho en la rama *(Script) [COBOL Script]* → *Properties* y en la pestaña *Compile*, en el apartado *Options*, escribimos *ALPHAL(WORD)* como se muestra en la siguiente ventana:



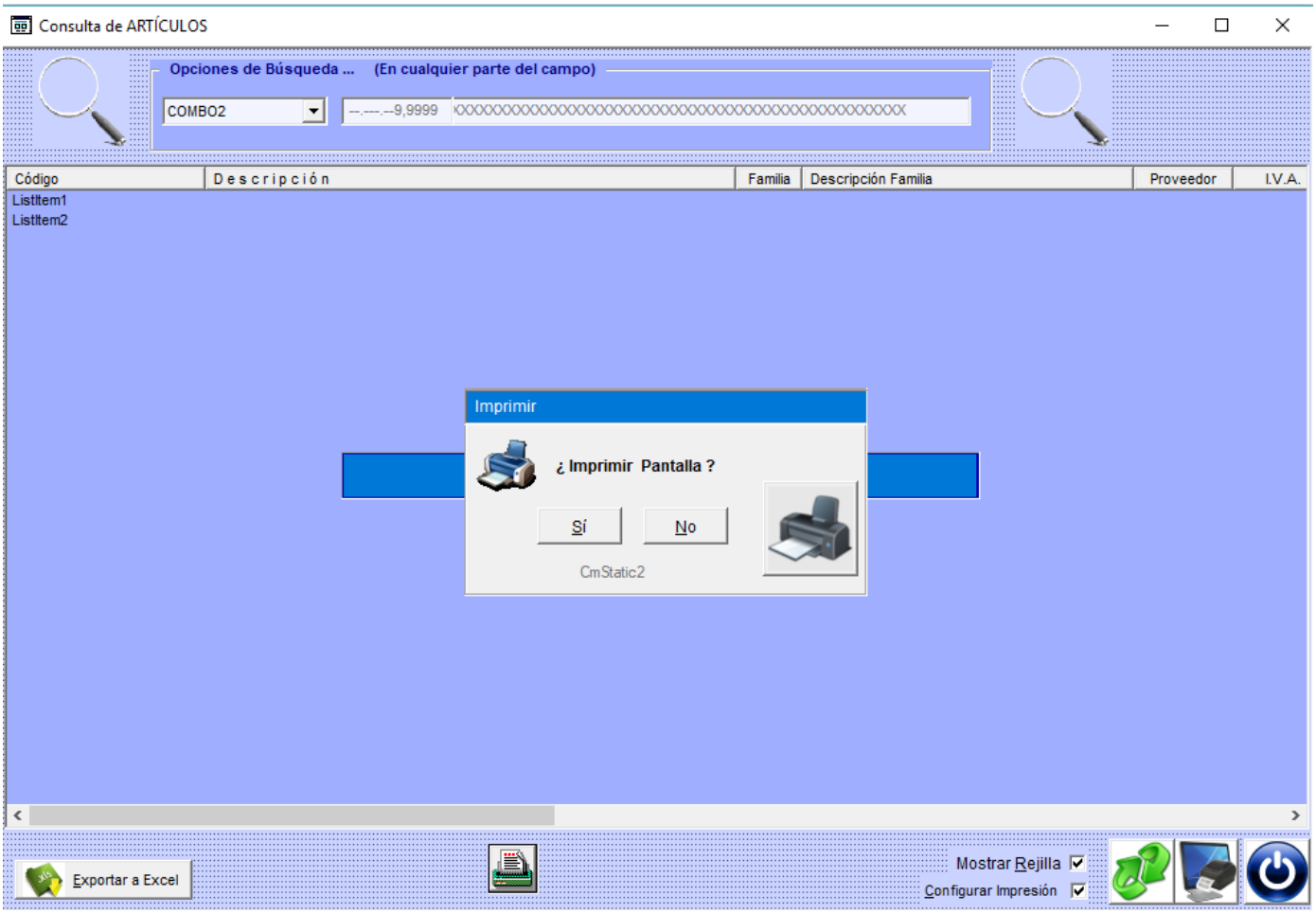
En ambos casos, podemos escribir más opciones para el compilador una detrás de la otra separadas por una **,** (coma). Yo en mi caso sólo utilizo esa y la opción **TEST**, que se utiliza para que el debugger pare dónde le indico, y poder ejecutar el programa linea a linea y ver donde está el posible error. (Si no pongo la opción TEST, no hay manera de que el debugger pare donde le indicamos).

El Programa "*CONSULTAS*"

Vamos a continuar, creando el programa de CONSULTAS, por lo que vamos a insertar un nuevo form al proyecto, para ello, en el árbol de la izquierda, hacemos click con el botón derecho sobre la rama ***M-ARTICULOS [Module]*** → *Create Form*, con los que nos insertará un nuevo form en nuestro proyecto.



Abrimos las propiedades del nuevo form y, siguiendo los pasos comentados anteriormente, lo configuramos, cambiándole el nombre por ***F-CON-ARTICULOS***. El mío tendrá un tamaño de 990x653 píxeles, y me ha quedado así



El *listView* tiene todos los campos del fichero de artículos y familias, excepto el campo ARTELI, y la "ventana" que se ve en medio, no es una ventana real, es un "montaje" hecho con un control *GroupBox*, un *Shape*, unos botones y varias cosas más.

La pantalla de CONSULTA, podremos buscar los artículos, por diversos campos, pudiendo estar el texto que deseamos buscar, en cualquier parte del campo. Además, podremos cambiar columnas de posición mediante drag 'n' drop, (gracias a la API de Windows), ordenar por columna, ascendente o descendente y seleccionar una ficha mediante doble click, etc.

Así que vamos a ver el código fuente de todo el form y controles, sobre todo para que veáis que, con un poco de imaginación y práctica, se puede hacer mucho en COBOL.

Programación del Form (Consultas)

Click con el botón derecho en el form y vamos a definir los datos comunes a todo el form

ENVIRONMENT DIVISION → SPECIAL-NAMES:

```
  DECIMAL-POINT IS COMMA.
```

ENVIRONMENT DIVISION → REPOSITORY.

```
     CLASS OLE       AS "*OLE"
     CLASS OLE-EX    AS "*OLE-EXCEPTION"
     CLASS COM-ARRAY AS "*COM-ARRAY".
```

ENVIRONMENT DIVISION → FILE CONTROL:

```
     SELECT OPTIONAL FAMILIAS
            ASSIGN TO "C:\Articulos\Datos\FAMILIAS.FIC"
            ORGANIZATION IS INDEXED
            ACCESS MODE  IS DYNAMIC
            RECORD KEY   IS FAMCOD
            FILE STATUS  IS STFAM
            LOCK IS AUTOMATIC.
```

DATA DIVISION → FILE SECTION:

```
 FD  FAMILIAS GLOBAL EXTERNAL
     LABEL RECORD IS STANDARD
     DATA RECORD IS REG-FAM.
 01  REG-FAM.
     03  FAMCOD                PIC XXX.
     03  FAMDES                PIC X(30).
```

DATA DIVISION → WORKING-STORAGE SECTION:

```
 01  ESTILO                    PIC S9(4) COMP-5   GLOBAL.
 01  RETORNO                   PIC S9(4) COMP-5   GLOBAL.
 01  WS-ATENCION               PIC X(17)          GLOBAL VALUE " ¡ Atención ... !".
 01  SALTO                     PIC X  VALUE X"0A" GLOBAL.
 01  AHIVA3T                   PIC XXX            GLOBAL EXTERNAL.
 01  AHIVA15T                  PIC X(15)          GLOBAL EXTERNAL.
*
 01  STFAM                     PIC XX             GLOBAL.
*
* --- * DECLARACIÓN DE LAS VARIABLES PARA SQL
*
                EXEC SQL
                     BEGIN DECLARE SECTION
                END-EXEC.
*
 01  REG-ART      GLOBAL.
     03  ARTCOD                PIC X(15).
     03  ARTDES                PIC X(60).
     03  ARTFAM                PIC XXX.
     03  ARTPRO                PIC S9(6).
     03  ARTIVA                PIC S999V99.
     03  ARTPRE1               PIC S9(7)V999.
     03  ARTPRE2               PIC S9(7)V999.
     03  ARTPRE3               PIC S9(7)V999.
```

```
      03  ARTOBS                PIC X(200).
      03  ARTACT                PIC X.
      03  ARTIMG1               PIC X(255).
      03  ARTIMG2               PIC X(255).
      03  ARTIMG3               PIC X(255).
      03  ARTIMG4               PIC X(255).
      03  ARTIMG5               PIC X(255).
      03  ARTIMG6               PIC X(255).
      03  ARTFEC                PIC X(8).
      03  ARTFEC1               PIC X(8).
      03  ARTELI                PIC X.      *>  Para  saber  si  un  artículo  ha  sido
eliminado (S/N).
       *>
       *> --- VARIABLES
       *>
 01  SQLSTATE                   PIC XXXXX   IS GLOBAL.
 01  SQLCODE                    PIC S9(9) COMP-5   GLOBAL.
 01  SQLMSG                     PIC X(600)     GLOBAL.
 01  CONTASQL                   PIC S9(9)   IS GLOBAL.  *>   CONTADOR   DE   NÚMERO   DE
REGISTROS
 01  WS-SQL                     PIC X(350)  IS GLOBAL.
*
            EXEC SQL
                 END DECLARE SECTION
            END-EXEC.
*
 01  CONTADOR                   PIC 9(7)   GLOBAL.
 01  WTEX                       PIC X(150) GLOBAL.
 01  WK-IDX-OLD                 PIC S9(9)  COMP-5 GLOBAL.
*
 01  WTIP_CON                   PIC 9      BINARY GLOBAL EXTERNAL.  *> Tipo  de  Búsqueda
inicial  0 = Código - 1 = Descripción
*
  #INCLUDE "C:\Manual\COPYS\EXTERNOS_PRINT.cop".
*
 01  WSCOMBO2.
      03                         PIC X(20) VALUE "Código".          *> 01
      03                         PIC X(20) VALUE "Descripción".     *> 02
      03                         PIC X(20) VALUE "Familia".         *> 03
      03                         PIC X(20) VALUE "Proveedor".       *> 04
      03                         PIC X(20) VALUE "I.V.A.".          *> 05
      03                         PIC X(20) VALUE "Precio 1".        *> 06
      03                         PIC X(20) VALUE "Precio 2".        *> 07
      03                         PIC X(20) VALUE "Precio 3".        *> 08
      03                         PIC X(20) VALUE "Fecha Alta".      *> 09
      03                         PIC X(20) VALUE "Fecha U. Modi.".  *> 10
      03                         PIC X(20) VALUE "Imagen 1".        *> 11
      03                         PIC X(20) VALUE "Observaciones".   *> 12
 01  WSCOM2 REDEFINES WSCOMBO2 IS GLOBAL.
      03  WSC2                  PIC X(20) OCCURS 12.
*
* Variables comunes para el GRID
*
 01  WS-HANDLE                    PIC S9(09)    COMP-5 GLOBAL VALUE 0.
 01  LVM-SETEXTENDEDLISTVIEWSTYLE PIC S9(09)    COMP-5 GLOBAL VALUE 4150.
       *>
       *> Variables para Drag'n'Drop de columnas
       *>
 01  WS-DRAG                      PIC S9(09)    COMP-5 GLOBAL.
 01  LVS-EX-HEADERDRAGDROP        PIC S9(09)    COMP-5 GLOBAL VALUE 16.
       *>
       *> Variables para colorear linea entera
       *>
 01  WS-LINEA-ENTERA              PIC S9(09)    COMP-5 GLOBAL.
 01  LVS-EX-FULLROWSELECT         PIC S9(09)    COMP-5 GLOBAL VALUE 32.
```

Eventos del Form

Evento KeyDown del form:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 LINKAGE         SECTION.
 01  POW-ARG-KEYCODE PIC S9(4) COMP-5.
 01  POW-ARG-SHIFT   PIC S9(4) COMP-5.
 PROCEDURE       DIVISION USING POW-ARG-KEYCODE POW-ARG-SHIFT.

     IF POW-ARG-KEYCODE = 27
        CALL "PARASALIR"
     END-IF.
```

Evento Opened:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  I                        PIC 99       BINARY.
*
 PROCEDURE       DIVISION.
 INICIO.
*
     INVOKE POW-SELF "ShowForm" USING POW-SWSHOWNORMAL.

     MOVE "hWnd" OF VIEW2 TO WS-HANDLE.
*
     CALL "SendMessageA" WITH STDCALL USING BY VALUE WS-HANDLE
                              BY VALUE 4150
                              BY VALUE 64
                              BY VALUE 64 *> COLUMNA 1, TIPO HiperLink
     CANCEL "SendMessageA".

     MOVE 32 TO WS-LINEA-ENTERA     *> SELECCIONAR LA LÍNEA ENTERA
     CALL "SendMessageA" WITH STDCALL USING BY VALUE WS-HANDLE
                                            BY VALUE LVM-SETEXTENDEDLISTVIEWSTYLE
                                            BY VALUE LVS-EX-FULLROWSELECT
                                            BY VALUE WS-LINEA-ENTERA
     CANCEL "SendMessageA".
*
     MOVE 16 TO WS-DRAG.            *> ARRASTRAR Y CAMBIAR UNA COLUMNA DE SITIO
     CALL "SendMessageA" WITH STDCALL USING BY VALUE WS-HANDLE
                                            BY VALUE LVM-SETEXTENDEDLISTVIEWSTYLE
                                            BY VALUE LVS-EX-HEADERDRAGDROP
                                            BY VALUE WS-DRAG
     CANCEL "SendMessageA".
*
     IF POW-CHECK OF CHECK4 = POW-ON
        CALL "SendMessageA" WITH STDCALL USING BY VALUE WS-HANDLE
                                 BY VALUE LVM-SETEXTENDEDLISTVIEWSTYLE
                                 BY VALUE 1
                                 BY VALUE 1 *> GRIDLINES - 1 = Si - 0 = No
        CANCEL "SendMessageA"
     ELSE
        CALL "SendMessageA" WITH STDCALL USING BY VALUE WS-HANDLE
                                 BY VALUE LVM-SETEXTENDEDLISTVIEWSTYLE
                                 BY VALUE 1
                                 BY VALUE 0 *> GRIDLINES - 1 = Si - 0 = No
        CANCEL "SendMessageA"
     END-IF.
*
```

```
* --- Llenar COMBO
*
     PERFORM VARYING I FROM 1 BY 1 UNTIL I > 12
             INVOKE COMBO2 "AddString" USING WSC2 (I)
     END-PERFORM.
     INVOKE POW-SELF "ShowForm" USING POW-SWSHOWNORMAL.
*
* --- Comprobar Tipo de Consulta
*
     IF AHIVA15T = SPACES
        CALL "CARGAR"
     ELSE
        CALL "CARGAR1"
     END-IF.

     INVOKE CAMPO2 "SetFocus".
     EXIT PROGRAM.
```

En este evento, hacemos varias llamadas a la API de Windows, (para “añadir” algunas opciones al control *listView*, porque las que trae por defecto en PWC, son bastante “justitas”), por lo que no olvidéis que hay que poner la directiva **ALPHAL(WORD)** antes de compilar.

Eventos de lo controles del Form

Evento *SelChange* de COMBO2:

```
 ENVIRONMENT      DIVISION.
 DATA             DIVISION.
 WORKING-STORAGE  SECTION.
 PROCEDURE        DIVISION.

     MOVE 0 TO "TextCase" OF CAMPO2.

     EVALUATE "ListIndex" OF COMBO2
              WHEN 1                  *> Por CÓDIGO
                   MOVE 1   TO "Visible"   OF CAMPO2
                   MOVE 1   TO "Enabled"   OF CAMPO2
                   MOVE 1   TO "TextCase"  OF CAMPO2
                   MOVE 0   TO "Enabled"   OF CAMPO1
                   MOVE 0   TO "Visible"   OF CAMPO1
                   MOVE 140 TO "Width"     OF CAMPO2
                   MOVE 15  TO "MaxLength" OF CAMPO2
                   INVOKE CAMPO2 "SetFocus"
                   EXIT PROGRAM
              WHEN 2                  *> Por DESCRIPCIÓN
                   MOVE 1   TO "Visible"   OF CAMPO2
                   MOVE 1   TO "Enabled"   OF CAMPO2
                   MOVE 0   TO "Enabled"   OF CAMPO1
                   MOVE 0   TO "Visible"   OF CAMPO1
                   MOVE 475 TO "Width"     OF CAMPO2
                   MOVE 40  TO "MaxLength" OF CAMPO2
                   INVOKE CAMPO2 "SetFocus"
                   EXIT PROGRAM
              WHEN 3                  *> Por FAMILIA
                   MOVE 0   TO "Visible"   OF CAMPO2
                   MOVE 0   TO "Enabled"   OF CAMPO2
                   MOVE 1   TO "Visible"   OF CAMPO1
                   MOVE 1   TO "Enabled"   OF CAMPO1
                   MOVE 85  TO "Width"     OF CAMPO1
                   MOVE 3   TO "MaxLength" OF CAMPO1
                   INVOKE CAMPO1 "SetFocus"
                   EXIT PROGRAM
              WHEN 4                  *> Por PROVEEDOR
                   MOVE 0   TO "Visible"   OF CAMPO2
                   MOVE 0   TO "Enabled"   OF CAMPO2
                   MOVE 1   TO "Visible"   OF CAMPO1
```

```
                MOVE 1   TO "Enabled"   OF CAMPO1
                MOVE 85  TO "Width"     OF CAMPO1
                MOVE 6   TO "MaxLength" OF CAMPO1
                INVOKE CAMPO1 "SetFocus"
                EXIT PROGRAM
            WHEN 5                  *> Por IVA
                MOVE 0   TO "Visible"   OF CAMPO2
                MOVE 0   TO "Enabled"   OF CAMPO2
                MOVE 1   TO "Visible"   OF CAMPO1
                MOVE 1   TO "Enabled"   OF CAMPO1
                MOVE 70  TO "Width"     OF CAMPO1
                MOVE 6   TO "MaxLength" OF CAMPO1
                INVOKE CAMPO1 "SetFocus"
                EXIT PROGRAM
            WHEN 6 THRU 8           *> Por PRECIO 1/2/3
                MOVE 0   TO "Visible"   OF CAMPO2
                MOVE 0   TO "Enabled"   OF CAMPO2
                MOVE 1   TO "Visible"   OF CAMPO1
                MOVE 1   TO "Enabled"   OF CAMPO1
                MOVE 85  TO "Width"     OF CAMPO1
                MOVE 10  TO "MaxLength" OF CAMPO1
                INVOKE CAMPO1 "SetFocus"
                EXIT PROGRAM
            WHEN 9 THRU 10          *> Por FECHA ALTA/ÚLTIMA MODIFICEACIÓN
                MOVE 0   TO "Visible"   OF CAMPO2
                MOVE 0   TO "Enabled"   OF CAMPO2
                MOVE 1   TO "Enabled"   OF CAMPO1
                MOVE 1   TO "Visible"   OF CAMPO1
                MOVE 70  TO "Width"     OF CAMPO2
                MOVE 8   TO "MaxLength" OF CAMPO2
                INVOKE CAMPO1 "SetFocus"
                EXIT PROGRAM
            WHEN 11                 *> Por IMAGEN 1
                MOVE 1   TO "Visible"   OF CAMPO2
                MOVE 1   TO "Enabled"   OF CAMPO2
                MOVE 0   TO "Visible"   OF CAMPO1
                MOVE 0   TO "Enabled"   OF CAMPO1
                MOVE 475 TO "Width"     OF CAMPO2
                MOVE 60  TO "MaxLength" OF CAMPO2
                INVOKE CAMPO2 "SetFocus"
                EXIT PROGRAM
            WHEN 12                 *> Por OBSERVACIONES
                MOVE 1   TO "Visible"   OF CAMPO2
                MOVE 1   TO "Enabled"   OF CAMPO2
                MOVE 0   TO "Enabled"   OF CAMPO1
                MOVE 0   TO "Visible"   OF CAMPO1
                MOVE 475 TO "Width"     OF CAMPO2
                MOVE 50  TO "MaxLength" OF CAMPO2
                INVOKE CAMPO2 "SetFocus"
                EXIT PROGRAM
            WHEN OTHER                  *> OTRO, se "setea" A 1
                MOVE 1   TO "Visible"   OF CAMPO2
                MOVE 1   TO "Enabled"   OF CAMPO2
                MOVE 0   TO "Enabled"   OF CAMPO1
                MOVE 0   TO "Visible"   OF CAMPO1
                MOVE 140 TO "Width"     OF CAMPO2
                MOVE 15  TO "MaxLength" OF CAMPO2
                MOVE 1   TO "ListIndex" OF COMBO2
                INVOKE CAMPO2 "SetFocus"
                EXIT PROGRAM
    END-EVALUATE.
```

Como podemos buscar tanto por campos numéricos como alfanuméricos, en este evento, mostramos un tipo de campo u otro, (*CAMPO1 y CAMPO2*), y además ajustamos el ancho del campo y la cantidad máxima de caracteres que se pueden introducir.

Evento *KeyPress* de CAMPO1:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 LINKAGE         SECTION.
 01  POW-ARG-KEYASCII PIC S9(4) COMP-5.
 PROCEDURE       DIVISION USING POW-ARG-KEYASCII.
*
     EVALUATE POW-ARG-KEYASCII

               WHEN 46         *> Cambiamos el punto del teclado numérico por una coma
                    MOVE 44 TO POW-ARG-KEYASCII

               WHEN 27         *> Tecla ESCAPE
                    CALL "PARASALIR"

     END-EVALUATE.
```

Evento *Return* de CAMPO1:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.

     CALL "CARGAR-N".
```

Evento *KeyPress* de CAMPO2:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 LINKAGE         SECTION.
 01  POW-ARG-KEYASCII PIC S9(4) COMP-5.
 PROCEDURE       DIVISION USING POW-ARG-KEYASCII.
*
     EVALUATE POW-ARG-KEYASCII

               WHEN 27         *> Tecla ESCAPE
                    CALL "PARASALIR"

     END-EVALUATE.
```

Evento *Return* de CAMPO2:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.

     CALL "CARGAR-T".
```

Evento *ColumnClick* de VIEW2:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 LINKAGE         SECTION.
 01   POW-ARG-COLUMNINDEX            PIC S9(9) COMP-5.
 PROCEDURE       DIVISION USING POW-ARG-COLUMNINDEX.

     IF "SortOrder" OF VIEW2 = 1
        MOVE 2 TO "SortOrder" OF VIEW2
     ELSE
        MOVE 1 TO "SortOrder" OF VIEW2
     END-IF.
*
     EVALUATE POW-ARG-COLUMNINDEX *> SortKind = 0 Texto - SortKind = 1 Numerico
              WHEN 1 THRU 4
                   MOVE 0                   TO "SortKind"   OF VIEW2
                   MOVE POW-ARG-COLUMNINDEX TO "SortColumn" OF VIEW2
              WHEN 5 THRU 9
                   MOVE 1                   TO "SortKind"   OF VIEW2
                   MOVE POW-ARG-COLUMNINDEX TO "SortColumn" OF VIEW2
              WHEN 10 THRU 14
                   MOVE 0                   TO "SortKind"   OF VIEW2
                   MOVE POW-ARG-COLUMNINDEX TO "SortColumn" OF VIEW2
     END-EVALUATE.
```

Evento *DblClick* de VIEW2:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01   POW-PCMLIST              OBJECT REFERENCE POW-CLISTITEM.
 01   WK-IDX                   PIC S9(9) COMP-5.
 PROCEDURE       DIVISION.

     MOVE "SelItemIndex" OF VIEW2 TO WK-IDX.

     IF WK-IDX < 1
        EXIT PROGRAM
     END-IF.

     MOVE "ListItems"(WK-IDX) OF VIEW2 TO POW-PCMLIST.
*
     MOVE "Text"(1) OF POW-PCMLIST TO AHIVA15T.

     INVOKE POW-SELF "CloseForm".
```

Evento *KeyPresss* de VIEW2:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 LINKAGE         SECTION.
 01   POW-ARG-KEYASCII PIC S9(4) COMP-5.
 PROCEDURE       DIVISION USING POW-ARG-KEYASCII.
*
     EVALUATE POW-ARG-KEYASCII

              WHEN 27          *> Tecla Esc
                   CALL "PARASALIR"
```

```
              WHEN 5          *> Tecla Ctrl. + E
                   CALL "A-EXCEL"

     END-EVALUATE.
```

Evento *Click* de BTEXCEL:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE DIVISION.
     CALL "A-EXCEL".
```

Evento *Click* de CHECK4:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.
*
     MOVE "hWnd" OF VIEW2 TO WS-HANDLE.
     IF POW-CHECK OF CHECK4 = POW-ON
        CALL "SendMessageA" WITH STDCALL USING BY VALUE WS-HANDLE
                                    BY VALUE LVM-SETEXTENDEDLISTVIEWSTYLE
                                    BY VALUE 1
                                    BY VALUE 1 *> GRIDLINES - 1 = Si - 0 = No
        CANCEL "SendMessageA"
     ELSE
        CALL "SendMessageA" WITH STDCALL USING BY VALUE WS-HANDLE
                                    BY VALUE LVM-SETEXTENDEDLISTVIEWSTYLE
                                    BY VALUE 1
                                    BY VALUE 0 *> GRIDLINES - 1 = Si - 0 = No
        CANCEL "SendMessageA"
     END-IF.

     EXIT PROGRAM.
```

Evento *Click* de botón BTRECARGAR:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.
     CALL "CARGAR".
```

Evento *Click* de botón BTLISTAR:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.
 INICIO.
     INVOKE POW-SELF "Alarm" USING POW-MBEXCLAMATION.
         ADD POW-DMICONQUESTION POW-DMYESNO POW-DMDEFBUTTON1
             GIVING ESTILO.
         INVOKE POW-SELF "DisplayMessage"
                USING "¿ Imprimir Pantalla ?"
                      WS-ATENCION
```

```
                    ESTILO
                RETURNING RETORNO
         END-INVOKE.
*
         IF RETORNO = POW-DMRYES

            IF POW-CHECK OF CmCheck1 = POW-ON
               PERFORM CARGAR-DATOS
               INVOKE POW-SELF "CallForm"
                      USING "F-PRINT" "M-PRINT"
               PERFORM VUELTA
               PERFORM IMPRIMIR
            ELSE
               PERFORM IMPRIMIR
            END-IF

         END-IF.
*
         EXIT PROGRAM.
 IMPRIMIR.
         INVOKE PRINT1 "PrintForm".
*
 CARGAR-DATOS.
     MOVE "Listado de ARTÍCULOS"    TO WL_TEXT.
     MOVE  "S"                TO WL_IMPFEC.     *> SI IMPRIMIR FECHA
     MOVE   2                 TO WL_PAG.  *> PÁGINA n DE n
     MOVE   2                 TO WL_ORI.  *> ORIENTACIÓN HORIZONTAL
     MOVE  90                 TO WL_ESCALA.
     MOVE   5                 TO WL_PAP   *> A4
     MOVE   1                 TO WL_ORIGEN.     *> BANDEJA POR DEFECTO
     MOVE   5                 TO WL_SUP.
     MOVE   5                 TO WL_INF.
     MOVE   5                 TO WL_IZQ.
     MOVE   5                 TO WL_DER.
     MOVE   0                 TO WL_ENC.
     MOVE ".\Listas\Combo1.TXT"   TO WL_RUTA1.
     MOVE ".\Listas\Combo2.TXT"   TO WL_RUTA2.
*
 VUELTA.
     IF WL_TEXT = SPACES      *> ÉSTE CAMPO NO PUEDE VALER ESPACIOS, YA QUE SE USA
COMO NOMBRE DEL FICHERO PARA IMPRIMIR
                         *> POR EJEMPLO PARA UN PDF (Listado de Clientes.pdf), POR LO
QUE SI VALE ESPACIOS, SE MUEVE
                         *> EL CARÁCTER Alt+255, QUE ES UN CARÁCTER INVISIBLE PARA
QUE NO DE ERROR.
        MOVE " " TO "HeadText" OF PRINT1
     ELSE
        MOVE WL_TEXT TO "HeadText" OF PRINT1
     END-IF.

     IF WL_IMPFEC = "S"
        MOVE 1 TO "PrintDate" OF PRINT1
     ELSE
        MOVE 0 TO "PrintDate" OF PRINT1
     END-IF.

     MOVE WL_PAG    TO "PageNoFormat"     OF PRINT1.
     MOVE WL_ORI    TO "PaperOrientation" OF PRINT1.

     IF WL_ESCALA = 0    *> ES EL ÚNICO VALOR QUE TENEMOS QUE COMPROBAR SI VALE 0,
                         *> PORQUE SI EN EL PROGRAMA LLAMADO, PULSAMOS SALIR,
                         *> NOS DEVUELVE 0 Y DARÍA UN ERROR
        MOVE  90    TO "Rate"             OF PRINT1
     ELSE
        MOVE WL_ESCALA TO "Rate"          OF PRINT1
     END-IF.
*
*     ----- TIPOS DE PAPEL -----
```

```
*
*     0 - Auto (depends on the target printer.) <--- VALOR DEVUELTO POR WL_PAP 1
*     (POW-DEFAULTSIZE)
*     1 - Letter 216x279 mm / Letter 8.5x11 inch      <--- VALOR DEVUELTO POR WL_PAP 2
*     (POW-PAPERLETTER)
*     5 - Legal 216x356 mm / Legal 8.5x14 inch  <--- VALOR DEVUELTO POR WL_PAP 3
*     (POW-PAPERLEGAL)
*     8 - A3 297x420 mm / A3 11.5x16.5 inch           <--- VALOR DEVUELTO POR WL_PAP 4
*     (POW-PAPERA3)
*     9 - A4 210x297 mm / A4 8.2x11.5 inch            <--- VALOR DEVUELTO POR WL_PAP 5
*     (POW-PAPERA4)
*     11 - A5 148x210 mm / A5 5.8x8.2 inch            <--- VALOR DEVUELTO POR WL_PAP 6
*     (POW-PAPERA5)
*     12 - B4 257x364 mm / B4 10x14.3 inch            <--- VALOR DEVUELTO POR WL_PAP 7
*     (POW-PAPERB4)
*     13 - B5 182x257 mm / B5 7x10 inch         <--- VALOR DEVUELTO POR WL_PAP 8
*     (POW-PAPERB5)

     EVALUATE WL_PAP

              WHEN 1
                   MOVE 0  TO "PaperType" OF PRINT1
              WHEN 2
                   MOVE 1  TO "PaperType" OF PRINT1
              WHEN 3
                   MOVE 5  TO "PaperType" OF PRINT1
              WHEN 4
                   MOVE 8  TO "PaperType" OF PRINT1
              WHEN 5
                   MOVE 9  TO "PaperType" OF PRINT1
              WHEN 6
                   MOVE 11 TO "PaperType" OF PRINT1
              WHEN 7
                   MOVE 12 TO "PaperType" OF PRINT1
              WHEN 8
                   MOVE 13 TO "PaperType" OF PRINT1
              WHEN OTHER
                   MOVE 0  TO "PaperType" OF PRINT1

     END-EVALUATE.

     SUBTRACT 1 FROM WL_ORIGEN.
     MOVE WL_ORIGEN TO "Source" OF PRINT1.

     MOVE WL_SUP TO "TopMargin"    OF PRINT1.
     MOVE WL_INF TO "BottomMargin" OF PRINT1.
     MOVE WL_IZQ TO "LeftMargin"   OF PRINT1.
     MOVE WL_DER TO "RightMargin"  OF PRINT1.
     MOVE WL_ENC TO "PunchMargin"  OF PRINT1.
```

Evento *Click* de botón BTSALIR

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.
     CALL "PARASALIR".
```

Evento *Click of GRUPO-PRN* de botón BTPRN-SI:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
```

```
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.
*
         IF POW-CHECK OF CmCheck1 = POW-ON
            PERFORM CARGAR-DATOS
            INVOKE POW-SELF "CallForm" USING "F-PRINT" "M-PRINT"
            PERFORM VUELTA
            PERFORM IMPRIMIR
         ELSE
               PERFORM IMPRIMIR
         END-IF.
*
     MOVE 0 TO "Visible" OF GRUPO-PRN.
     EXIT PROGRAM.
 IMPRIMIR.
         INVOKE PRINT1 "PrintForm".
*
 CARGAR-DATOS.
     MOVE "PrinterName" OF PRINT1    TO WL_PRINT.
     MOVE "Consulta de ARTÍCULOS"    TO WL_TEXT.
     MOVE  "S"                       TO WL_IMPFEC.      *> SI IMPRIMIR FECHA
     MOVE   2                        TO WL_PAG.         *> PÁGINA n DE n
     MOVE   2                        TO WL_ORI.         *> ORIENTACIÓN HORIZONTAL
     MOVE  100                       TO WL_ESCALA.
     MOVE   5                        TO WL_PAP          *> A4
     MOVE   1                        TO WL_ORIGEN.      *> BANDEJA POR DEFECTO
     MOVE   5                        TO WL_SUP.
     MOVE   5                        TO WL_INF.
     MOVE   5                        TO WL_IZQ.
     MOVE   5                        TO WL_DER.
     MOVE   0                        TO WL_ENC.
     MOVE ".\Listas\Combo1.TXT"      TO WL_RUTA1.
     MOVE ".\Listas\Combo2.TXT"      TO WL_RUTA2.
*
 VUELTA.
     IF WL_TEXT = SPACES      *> ÉSTE CAMPO NO PUEDE VALER ESPACIOS, YA QUE SE USA
COMO NOMBRE DEL FICHERO PARA IMPRIMIR
                              *> POR EJEMPLO PARA UN PDF (Listado de Clientes.pdf), POR LO
QUE SI VALE ESPACIOS, SE MUEVE
                              *> EL CARÁCTER Alt+255, QUE ES UN CARÁCTER INVISIBLE PARA
QUE NO DE ERROR.
        MOVE " " TO "HeadText" OF PRINT1
     ELSE
        MOVE WL_TEXT TO "HeadText" OF PRINT1
     END-IF.

     IF WL_IMPFEC = "S"
        MOVE 1 TO "PrintDate" OF PRINT1
     ELSE
        MOVE 0 TO "PrintDate" OF PRINT1
     END-IF.

     MOVE WL_PAG    TO "PageNoFormat"     OF PRINT1.
     MOVE WL_ORI    TO "PaperOrientation" OF PRINT1.

     IF WL_ESCALA = 0    *> ES EL ÚNICO VALOR QUE TENEMOS QUE COMPROBAR SI VALE 0,
                         *> PORQUE SI EN EL PROGRAMA LLAMADO, PULSAMOS SALIR,
                         *> NOS DEVUELVE 0 Y DARÍA UN ERROR
        MOVE 100       TO "Rate"          OF PRINT1
     ELSE
        MOVE WL_ESCALA TO "Rate"          OF PRINT1
     END-IF.
*
*     ----- TIPOS DE PAPEL -----
*
*     0 - Auto (depends on the target printer.) <--- VALOR DEVUELTO POR WL_PAP 1
*     (POW-DEFAULTSIZE)
*     1 - Letter 216x279 mm / Letter 8.5x11 inch      <--- VALOR DEVUELTO POR WL_PAP 2
```

```
*      (POW-PAPERLETTER)
*      5 - Legal 216x356 mm / Legal 8.5x14 inch  <--- VALOR DEVUELTO POR WL_PAP 3
*      (POW-PAPERLEGAL)
*      8 - A3 297x420 mm / A3 11.5x16.5 inch           <--- VALOR DEVUELTO POR WL_PAP 4
*      (POW-PAPERA3)
*      9 - A4 210x297 mm / A4 8.2x11.5 inch            <--- VALOR DEVUELTO POR WL_PAP 5
*      (POW-PAPERA4)
*      11 - A5 148x210 mm / A5 5.8x8.2 inch            <--- VALOR DEVUELTO POR WL_PAP 6
*      (POW-PAPERA5)
*      12 - B4 257x364 mm / B4 10x14.3 inch            <--- VALOR DEVUELTO POR WL_PAP 7
*      (POW-PAPERB4)
*      13 - B5 182x257 mm / B5 7x10 inch         <--- VALOR DEVUELTO POR WL_PAP 8
*      (POW-PAPERB5)

     EVALUATE WL_PAP

              WHEN 1
                   MOVE 0  TO "PaperType" OF PRINT1
              WHEN 2
                   MOVE 1  TO "PaperType" OF PRINT1
              WHEN 3
                   MOVE 5  TO "PaperType" OF PRINT1
              WHEN 4
                   MOVE 8  TO "PaperType" OF PRINT1
              WHEN 5
                   MOVE 9  TO "PaperType" OF PRINT1
              WHEN 6
                   MOVE 11 TO "PaperType" OF PRINT1
              WHEN 7
                   MOVE 12 TO "PaperType" OF PRINT1
              WHEN 8
                   MOVE 13 TO "PaperType" OF PRINT1
              WHEN OTHER
                   MOVE 0  TO "PaperType" OF PRINT1

     END-EVALUATE.

     SUBTRACT 1 FROM WL_ORIGEN.
     MOVE WL_ORIGEN TO "Source" OF PRINT1.

     MOVE WL_SUP TO "TopMargin"    OF PRINT1.
     MOVE WL_INF TO "BottomMargin" OF PRINT1.
     MOVE WL_IZQ TO "LeftMargin"   OF PRINT1.
     MOVE WL_DER TO "RightMargin"  OF PRINT1.
     MOVE WL_ENC TO "PunchMargin"  OF PRINT1.
```

Evento *Click of GRUPO-PRN* de botón BTPRN-NO:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.
     MOVE 0 TO "Visible" OF GRUPO-PRN.
     INVOKE BTLISTAR "SetFocus".
     EXIT PROGRAM.
```

Evento *Click of GRUPO-PRN* de botón BTPRN-SI:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.
     INVOKE PRINT1 "SetPrinter".
     MOVE "PrinterName" OF PRINT1 TO "Caption" OF INFO-PRN OF GRUPO-PRN.
```

Subrutinas programa CONSULTAS

Subrutina "A-EXCEL"

```
*
*
* ----- Mil Gracias por el código RAIPINTO
*
*
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  EXCEL              OBJECT REFERENCE OLE.
 01  WORKBOOK           OBJECT REFERENCE OLE.
 01  SHEETS             OBJECT REFERENCE OLE.
 01  WORKSHEET          OBJECT REFERENCE OLE.
 01  CELL               OBJECT REFERENCE OLE.
 01  COLUMNS            OBJECT REFERENCE OLE.
 01  WCOLUMN            OBJECT REFERENCE OLE.
 01  OBJRANGE           OBJECT REFERENCE OLE.
 01  FITRANGE           OBJECT REFERENCE OLE.
*
 01  ARRAYOBJ OBJECT REFERENCE COM-ARRAY.
 01  LONG-INT-TYPE                  PIC S9(9) COMP-5 VALUE 12.
 01  ARRAY-DIMENSION                PIC S9(9) COMP-5 VALUE 2.
 01  AXIS-1                         PIC S9(9) COMP-5 VALUE 22.
 01  AXIS-2                         PIC S9(9) COMP-5 VALUE 22.
*
 01  APPLICATION                    PIC X(20) VALUE "EXCEL.APPLICATION".
 01  OLE-TRUE                       PIC 1(1)  BIT VALUE B"1".
 01  FILLER                         PIC 1(7)  BIT.
 01  OLE-FALSE                      PIC 1(1)  BIT VALUE B"0".
 01  FILLER                         PIC 1(7)  BIT.
 01  ARRAY-ROW                      PIC S9(9) COMP-5.
 01  ARRAY-COL                      PIC S9(9) COMP-5.
 01  VAL                            PIC X(256).
 01  S-INDEX                        PIC S9(4) COMP-5 VALUE 1.
 01  LINHA                          PIC X(1024).
 01  OLE-ERR-METHOD                 PIC X(256).
 01  OLE-ERR-INFO.
     03  OLE-ERR-TYPE           PIC X(001).
     03  OLE-ERR-WCODE          PIC X(002).
     03  ROLE-ERR-WCODE REDEFINES OLE-ERR-WCODE  PIC S9(04) COMP-5.
     03  OLE-ERR-SCODE          PIC X(004).
     03  ROLE-ERR-SCODE REDEFINES OLE-ERR-SCODE PIC S9(09) COMP-5.
*
 01  XARRCOL                    PIC X(026)  VALUE "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
 01  ARRCOLS  REDEFINES XARRCOL.
     03   ARRCOL    OCCURS 26   PIC X(001).
*
 01  FORMATNUM                PIC X(9)  VALUE "#.##0,00".
 01  FORMATDATE               PIC X(12) VALUE "aaaa-mm-dd".
 01  NRCOLS                   PIC S9(9) COMP-5   VALUE 1.
 01  INIROW                   PIC S9(9) COMP-5   VALUE 2.
 01  INICOL                   PIC S9(9) COMP-5   VALUE 1.
 01  ENDROW                   PIC S9(9) COMP-5   VALUE 2.
 01  ENDCOL                   PIC S9(9) COMP-5   VALUE 22.
 01  INIRANGE                 PIC X(19)          VALUE " A0000002:AZ0000002".
 01  ROWVAL                   PIC 9(007).
 01  COLVAL                   PIC 9(007).
 01  PICSTRING                PIC X(020).
 01  PICDATE                  PIC X(10).
*
 01  IDX1                     PIC S9(5) COMP-5.
 01  IDX2                     PIC S9(5) COMP-5.
 01  IDX3                     PIC S9(5) COMP-5.
```

```
 01  IDX4                    PIC S9(5) COMP-5.
 01  IDX5                    PIC S9(5) COMP-5.
 01  WSLONG                  PIC S9(5) COMP-5.
*
 PROCEDURE DIVISION.
 DECLARATIVES.
 OLE-ERRO SECTION.
     USE AFTER EXCEPTION OLE-EX.
*
 END DECLARATIVES.
*
 MAIN SECTION.
 MAIN-00.
           IF "COUNT" OF View2 = ZERO
              GO TO MAIN-99
           END-IF.

           MOVE 11 TO "MousePointer" OF POW-SELF.
           MOVE 1  TO WSLONG.
           MOVE 1  TO S-INDEX.

           INVOKE OLE "CREATE-OBJECT" USING APPLICATION
                  RETURNING EXCEL
           END-INVOKE.

           INVOKE EXCEL "GET-WORKBOOKS"
                  RETURNING WORKBOOK
           END-INVOKE.

           INVOKE WORKBOOK "ADD"
                  RETURNING WORKBOOK
           END-INVOKE.

           INVOKE WORKBOOK "GET-WORKSHEETS"
                  RETURNING SHEETS
           END-INVOKE.

           INVOKE SHEETS "GET-ITEM"
                  USING S-INDEX
                  RETURNING WORKSHEET
           END-INVOKE.
*
           MOVE "ColumnCount" OF View2    TO IDX1.
*
           MOVE 1 TO IDX5.

           PERFORM VARYING IDX2 FROM 1 BY 1 UNTIL IDX2 > IDX1
                   IF "Width" OF "Columns"(IDX2) OF View2 > 5
                       MOVE "HeaderText" OF "Columns"(IDX2) OF View2 TO VAL
                       MOVE  WSLONG TO ARRAY-ROW
                       MOVE IDX5 TO ARRAY-COL
                       INVOKE WORKSHEET "GET-CELLS"
                              USING ARRAY-ROW ARRAY-COL
                              RETURNING CELL
                       END-INVOKE
                       INVOKE CELL "SET-VALUE"
                              USING VAL
                       END-INVOKE
                       ADD 1                              TO IDX5
                   END-IF
           END-PERFORM.
*
           ADD 1                          TO WSLONG.
           MOVE "Count" OF View2          TO IDX1.
           MOVE "ColumnCount" OF View2    TO IDX2.
*
           MOVE IDX1 TO AXIS-1.
```

```
           MOVE IDX2 TO AXIS-2.

           INVOKE COM-ARRAY "NEW"
                  USING LONG-INT-TYPE
                        ARRAY-DIMENSION
                        AXIS-1
                        AXIS-2
                  RETURNING ARRAYOBJ
           END-INVOKE.

           INVOKE POW-SELF "THRUEVENTS".
*
*  VAI CARREGAR O ARRAY BIDINENSIONAL OLE-ARRAY A PARTIR DA LISTVIEW
*
           PERFORM VARYING IDX3 FROM 1 BY 1 UNTIL IDX3 > IDX1
                   MOVE 1 TO IDX5
                   PERFORM VARYING IDX4 FROM 1 BY 1 UNTIL IDX4 > IDX2
                           IF "Width" OF "Columns"(IDX4) OF View2 > 5
                              MOVE "Text"(IDX4) OF "ListItems"(IDX3) OF View2   TO
LINHA
                              INVOKE ARRAYOBJ "SET-DATA"
                                     USING LINHA IDX3 IDX5
                              END-INVOKE
                              ADD 1 TO IDX5
                           END-IF
                   END-PERFORM
                   IF IDX3 = 5000 OR = 10000 OR = 15000 OR = 20000 OR = 250000 OR =
30000
                      OR = 35000 OR = 40000
                         INVOKE POW-SELF "THRUEVENTS"
                   END-IF
           END-PERFORM.
*
************  CALCULAR O "RANGE" DA FOLHA TODA PARA ENVIAR O ARRAY DUMA SO VEZ
*  A PRIMEIRA CELULA É SEMPRE "A00002"
           MOVE 1 TO ROWVAL.
           ADD  1 TO ROWVAL .
           MOVE ROWVAL TO INIRANGE(3:7).

           EVALUATE IDX2
                    WHEN 183 THRU 208
                         SUBTRACT 182 FROM IDX2 GIVING COLVAL
                         MOVE "G"             TO INIRANGE(11:1)
                         MOVE ARRCOL(COLVAL) TO INIRANGE(12:1)
                    WHEN 157 THRU 182
                         SUBTRACT 156 FROM IDX2 GIVING COLVAL
                         MOVE "F"             TO INIRANGE(11:1)
                         MOVE ARRCOL(COLVAL) TO INIRANGE(12:1)
                    WHEN 131 THRU 156
                         SUBTRACT 130 FROM IDX2 GIVING COLVAL
                         MOVE "E"             TO INIRANGE(11:1)
                         MOVE ARRCOL(COLVAL) TO INIRANGE(12:1)
                    WHEN 105 THRU 130
                         SUBTRACT 104 FROM IDX2 GIVING COLVAL
                         MOVE "D"             TO INIRANGE(11:1)
                         MOVE ARRCOL(COLVAL) TO INIRANGE(12:1)
                    WHEN 79 THRU 104
                         SUBTRACT 78 FROM IDX2 GIVING COLVAL
                         MOVE "C"             TO INIRANGE(11:1)
                         MOVE ARRCOL(COLVAL) TO INIRANGE(12:1)
                    WHEN 53 THRU 78
                         SUBTRACT 52 FROM IDX2 GIVING COLVAL
                         MOVE "B"             TO INIRANGE(11:1)
                         MOVE ARRCOL(COLVAL) TO INIRANGE(12:1)
                    WHEN 27 THRU 52
                         SUBTRACT 26 FROM IDX2 GIVING COLVAL
                         MOVE "A"             TO INIRANGE(11:1)
                         MOVE ARRCOL(COLVAL) TO INIRANGE(12:1)
```

```
                    WHEN 1 THRU 26
                         MOVE " "          TO INIRANGE(11:1)
                         MOVE ARRCOL(IDX2) TO INIRANGE(12:1)
           END-EVALUATE

           MOVE IDX1 TO ROWVAL.
           ADD 1 TO ROWVAL.
           MOVE ROWVAL TO INIRANGE(13:7).
           INVOKE WORKSHEET "GET-RANGE"
                  USING INIRANGE
                  RETURNING OBJRANGE
           END-INVOKE.
           INVOKE OBJRANGE "SET-VALUE"
                  USING ARRAYOBJ
           END-INVOKE.
*
 MAIN-80.
*
* UTILIZANDO O RANGE QUE FOI DAS COLUNAS/LINHAS UTILIZADAS VAI FAZER O AUTOFIT
(ALARGAR AS COLUNAS)
*
            INVOKE WORKSHEET "GET-USEDRANGE"
                   RETURNING OBJRANGE
            END-INVOKE.
            INVOKE OBJRANGE "GET-ENTIRECOLUMN"
                   RETURNING FITRANGE
            END-INVOKE.
            INVOKE FITRANGE "AUTOFIT".
            MOVE 0 TO "MousePointer" OF POW-SELF.
*  DESTRUIR OS OBJECTOS UTILIZADOS PARA LIBERTAR MEMORIA
            INVOKE EXCEL "SET-VISIBLE" USING OLE-TRUE.
*           INVOKE EXCEL "QUIT".
            SET CELL      TO NULL.
            SET COLUMNS   TO NULL.
            SET FITRANGE  TO NULL.
            SET OBJRANGE  TO NULL.
            SET WORKSHEET TO NULL.
            SET SHEETS    TO NULL.
            SET WORKBOOK  TO NULL.
            SET EXCEL     TO NULL.
 MAIN-99.
            EXIT PROGRAM.
```

Subrutina "CARGAR"

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01   POW-PCMLIST              OBJECT REFERENCE POW-CLISTITEM.
 01   WK-IDX                   PIC S9(9) COMP-5.
 01   TAM                      PIC 999 VALUE 0.
 01   WTEX                     PIC X(250) VALUE SPACES.
*
 01   CONTA-EDI                PIC ZZZZZ9.
 01   CONTA                    PIC 9(4).
*
 01   FECHA.
      03   AA                  PIC XXXX.
      03   MM                  PIC XX.
      03   DD                  PIC XX.
 01   FECHA1.
      03   DD                  PIC 99.
      03   MM                  PIC 99.
      03   AA                  PIC 9999.
 PROCEDURE       DIVISION.
```

```
*
* ----- * -----
*
     MOVE SPACES TO AHIVA15T.
     MOVE ZEROES TO "Text"      OF CAMPO1.
     MOVE SPACES TO "Text"      OF CAMPO2.
     MOVE 1      TO "Visible"   OF CAMPO2.
     MOVE 0      TO "Visible"   OF CAMPO1.
     MOVE 0      TO "Enabled"   OF CAMPO1.
     MOVE 1      TO "ListIndex" OF COMBO2.
*
     MOVE 0 TO CONTASQL WK-IDX CONTA.
*
* ----- Contamos la cantidad de registros para el progressbar
*

        EXEC SQL
             SELECT COUNT(ARTCOD)
                    FROM ARTICULOS
                         WHERE ARTELI <> "S"
                    INTO :CONTASQL
        END-EXEC.

     IF CONTASQL = 0
        ADD POW-DMICONERROR POW-DMOK GIVING ESTILO
        INVOKE POW-SELF "DisplayMessage"
               USING "*Error: El fichero está VACÍO ..."
                     WS-ATENCION
                     ESTILO
        END-INVOKE
        INVOKE POW-SELF "CloseForm"
        EXIT PROGRAM
     ELSE
        MOVE SPACES TO WTEX
        MOVE CONTASQL TO CONTA-EDI
        STRING CONTA-EDI
               " Artículos. Click en columna para ordenar."
                DELIMITED BY SIZE INTO WTEX
        END-STRING
        MOVE WTEX TO "ToolTipText" OF VIEW2
     END-IF.

         MOVE 1        TO "Visible"      OF VIEW2.
         MOVE 1        TO "Enabled"      OF VIEW2.
         INVOKE VIEW2 "Clear".
         MOVE 11       TO "MousePointer" OF POW-SELF.
         MOVE 1        TO "Visible"      OF PROGRESO.
         MOVE 0        TO "Min"          OF PROGRESO.
         MOVE CONTASQL TO "Max"          OF PROGRESO.
*
*
 LEO.
     MOVE SPACES TO WS-SQL.

     STRING "SELECT "
            "ARTCOD, "
                "ARTDES, "
                "ARTFAM, "
                "ARTPRO, "
                "ARTIVA, "
                "ARTPRE1, "
                "ARTPRE2, "
                "ARTPRE3, "
                "ARTOBS, "
                "ARTACT, "
                "ARTIMG1, "
                "ARTFEC, "
                "ARTFEC1 "
```

```
           "FROM ARTICULOS WHERE "
           "ARTELI <> 'S'"
           DELIMITED BY SIZE INTO WS-SQL *>WITH POINTER TAM
      END-STRING.

      EXEC SQL
           DECLARE cursor_con CURSOR FOR CONSULTA
      END-EXEC.

      EXEC SQL
           PREPARE CONSULTA FROM :WS-SQL
      END-EXEC.

      EXEC SQL
           OPEN cursor_con
      END-EXEC.

      PERFORM CARGAR THRU FIN-CARGAR
              UNTIL SQLSTATE = "02000".

      EXEC SQL
           COMMIT
      END-EXEC.

*
 FIN-LEO.
      MOVE 0 TO "Visible"      OF PROGRESO.
      MOVE 0 TO "MousePointer" OF POW-SELF.
          MOVE 1 TO "Enabled"      OF View2.
          MOVE 1 TO "Visible"      OF View2.
      MOVE 1 TO "Enabled"      OF BTLISTAR
*
      INVOKE COMBO2 "SetFocus".
      EXIT PROGRAM.
*
       CARGAR.

      EXEC SQL
           FETCH cursor_con
                 INTO
                    :ARTCOD,
                        :ARTDES,
                        :ARTFAM,
                        :ARTPRO,
                        :ARTIVA,
                        :ARTPRE1,
                        :ARTPRE2,
                        :ARTPRE3,
                        :ARTOBS,
                        :ARTACT,
                        :ARTIMG1,
                        :ARTFEC,
                        :ARTFEC1
      END-EXEC.

      IF SQLSTATE = "02000"
         GO FIN-CARGAR
      END-IF.

      INVOKE VIEW2 "Add" USING 1 1 RETURNING WK-IDX.

      IF WK-IDX > 0 THEN
         MOVE "ListItems"(WK-IDX) OF VIEW2 TO POW-PCMLIST
      END-IF.

      MOVE ARTCOD         TO "Text"     (01) OF POW-PCMLIST.
      MOVE ARTDES         TO "Text"     (02) OF POW-PCMLIST.
```

```
     MOVE ARTFAM        TO "Text"    (03) OF POW-PCMLIST.

     MOVE ARTFAM TO FAMCOD.
     READ FAMILIAS NO LOCK
          INVALID KEY
                  MOVE " No existe en fichero ..." TO "Text"    (04)    OF POW-PCMLIST
          NOT INVALID KEY
                  MOVE FAMDES                      TO "Text"    (04)    OF POW-PCMLIST
     END-READ.

     MOVE ARTPRO        TO "Numeric" (05) OF POW-PCMLIST.
     MOVE ARTIVA        TO "Numeric" (06) OF POW-PCMLIST.
     MOVE ARTPRE1       TO "Numeric" (07) OF POW-PCMLIST.
     MOVE ARTPRE2       TO "Numeric" (08) OF POW-PCMLIST.
     MOVE ARTPRE3       TO "Numeric" (09) OF POW-PCMLIST.

     MOVE ARTFEC        TO FECHA.
     MOVE CORR FECHA    TO FECHA1.
     MOVE FECHA1        TO "Text"    (10)       OF POW-PCMLIST.

     MOVE ARTFEC1       TO FECHA.
     MOVE CORR FECHA    TO FECHA1.
     MOVE FECHA1        TO "Text"    (11)       OF POW-PCMLIST.

     MOVE ARTIMG1       TO "Text"    (12) OF POW-PCMLIST.
     MOVE ARTOBS        TO "Text"    (13) OF POW-PCMLIST.
     MOVE ARTACT        TO "Text"    (14) OF POW-PCMLIST.

     INVOKE PROGRESO "ProgressStep".
     MOVE WK-IDX TO CONTA.

     IF CONTA = 1000
        OR = 2000
           OR = 3000
              OR = 4000
                 OR = 5000
                    OR = 6000
                       OR = 7000
                          OR = 8000
                             OR = 9000
                                OR = 9999
                                   INVOKE POW-SELF "ThruEvents"
     END-IF.
*
 FIN-CARGAR.
     EXIT.
```

Subrutina "CARGAR1"

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  POW-PCMLIST             OBJECT REFERENCE POW-CLISTITEM.
 01  WK-IDX                  PIC S9(9)  COMP-5.
 01  TAM                     PIC 999    COMP-5 VALUE 0.
 01  WTEX                    PIC X(250)        VALUE SPACES.
*
 01  CONTA-EDI               PIC ZZZZZ9.
 01  CONTA                   PIC 9(4)   COMP-5.
*
 01  FECHA.
     03  AA                  PIC XXXX.
     03  MM                  PIC XX.
     03  DD                  PIC XX.
 01  FECHA1.
```

```
     03  DD                   PIC 99.
     03  MM                   PIC 99.
     03  AA                   PIC 9999.
*
 PROCEDURE       DIVISION.
*
* ----- * -----
*
     MOVE ZEROES TO "Text"      OF CAMPO1.
     MOVE SPACES TO "Text"      OF CAMPO2.
     MOVE 1      TO "Visible"   OF CAMPO2.
     MOVE 0      TO "Visible"   OF CAMPO1.
     MOVE 0      TO "Enabled"   OF CAMPO1.
     MOVE 1      TO "ListIndex" OF COMBO2.
     INVOKE VIEW2 "Clear".
*
     MOVE 0 TO CONTASQL WK-IDX CONTA.
     MOVE AHIVA15T TO ARTCOD.
     MOVE AHIVA15T TO "Text" OF CAMPO2.
     MOVE SPACES TO AHIVA15T.
*
* ----- Contamos la cantidad de registros para el progressbar
*

        EXEC SQL
             SELECT COUNT(ARTCOD)
                    FROM ARTICULOS
                         WHERE ARTCOD LIKE CONCAT('%',TRIM(:ARTCOD),'%')
                               AND ARTELI <> 'S'
                    INTO :CONTASQL
        END-EXEC.

     IF CONTASQL = 0
        ADD POW-DMICONERROR POW-DMOK GIVING ESTILO
        INVOKE POW-SELF "DisplayMessage"
               USING "*Error: No hay coincidencias ..."
                     WS-ATENCION
                     ESTILO
        END-INVOKE
        MOVE 1 TO "Enabled" OF VIEW2
        MOVE 0 TO "Visible" OF PROGRESO
        INVOKE COMBO2 "SetFocus"
        EXIT PROGRAM
     ELSE
        MOVE SPACES TO WTEX
        MOVE CONTASQL TO CONTA-EDI
        STRING CONTA-EDI
               " Artículos. Click en columna para ordenar."
                DELIMITED BY SIZE INTO WTEX
        END-STRING
        MOVE WTEX TO "ToolTipText" OF VIEW2
     END-IF.

         MOVE 1         TO "Visible"      OF VIEW2.
         MOVE 1         TO "Enabled"      OF VIEW2.
         MOVE 11        TO "MousePointer" OF POW-SELF.
         MOVE 1         TO "Visible"      OF PROGRESO.
         MOVE 0         TO "Min"          OF PROGRESO.
         MOVE CONTASQL TO "Max"           OF PROGRESO.
*
*
 LEO.
     MOVE SPACES TO WS-SQL.

     STRING "SELECT "
            "ARTCOD, "
                "ARTDES, "
                "ARTFAM, "
```

```
                "ARTPRO, "
                "ARTIVA, "
                "ARTPRE1, "
                "ARTPRE2, "
                "ARTPRE3, "
                "ARTOBS, "
                "ARTACT, "
                "ARTIMG1, "
                "ARTFEC, "
                "ARTFEC1 "
            "FROM ARTICULOS WHERE "
            "ARTCOD LIKE CONCAT('%',TRIM('"
            ARTCOD
            "'),'%') AND ARTELI <> 'S'"
            DELIMITED BY SIZE INTO WS-SQL *>WITH POINTER TAM
      END-STRING.

      COMPUTE TAM = FUNCTION STORED-CHAR-LENGTH (WS-SQL) + 2.

      EXEC SQL
           DECLARE cursor_1 CURSOR FOR CONSULTA
      END-EXEC.

      EXEC SQL
           PREPARE CONSULTA FROM :WS-SQL
      END-EXEC.

      EXEC SQL
           OPEN cursor_1
      END-EXEC.

      PERFORM CARGAR THRU FIN-CARGAR
              UNTIL SQLSTATE = "02000".

      EXEC SQL
           COMMIT
      END-EXEC.

*
 FIN-LEO.
      MOVE 0 TO "Visible"      OF PROGRESO.
      MOVE 0 TO "MousePointer" OF POW-SELF.
          MOVE 1 TO "Enabled"      OF View2.
          MOVE 1 TO "Visible"      OF View2.
      MOVE 1 TO "Enabled"      OF BTLISTAR.
*
      INVOKE COMBO2 "SetFocus".
      EXIT PROGRAM.
*
       CARGAR.
      EXEC SQL
           FETCH cursor_1
                 INTO
                    :ARTCOD,
                        :ARTDES,
                        :ARTFAM,
                        :ARTPRO,
                        :ARTIVA,
                        :ARTPRE1,
                        :ARTPRE2,
                        :ARTPRE3,
                        :ARTOBS,
                        :ARTACT,
                        :ARTIMG1,
                        :ARTFEC,
                        :ARTFEC1
      END-EXEC.
```

```
     IF SQLSTATE = "02000"
        GO FIN-CARGAR
     END-IF.

     INVOKE VIEW2 "Add" USING 1 1 RETURNING WK-IDX.

     IF WK-IDX > 0 THEN
        MOVE "ListItems"(WK-IDX) OF VIEW2 TO POW-PCMLIST
     END-IF.

     MOVE ARTCOD         TO "Text"    (01) OF POW-PCMLIST.
     MOVE ARTDES         TO "Text"    (02) OF POW-PCMLIST.
     MOVE ARTFAM         TO "Text"    (03) OF POW-PCMLIST.

     MOVE ARTFAM TO FAMCOD.
     READ FAMILIAS NO LOCK
          INVALID KEY
                  MOVE " No existe en fichero ..." TO "Text"    (04)    OF POW-PCMLIST
          NOT INVALID KEY
                  MOVE FAMDES                      TO "Text"    (04)    OF POW-PCMLIST
     END-READ.

     MOVE ARTPRO         TO "Numeric" (05) OF POW-PCMLIST.
     MOVE ARTIVA         TO "Numeric" (06) OF POW-PCMLIST.
     MOVE ARTPRE1        TO "Numeric" (07) OF POW-PCMLIST.
     MOVE ARTPRE2        TO "Numeric" (08) OF POW-PCMLIST.
     MOVE ARTPRE3        TO "Numeric" (09) OF POW-PCMLIST.

     MOVE ARTFEC     TO FECHA.
     MOVE CORR FECHA TO FECHA1.
     MOVE FECHA1           TO "Text"    (10)       OF POW-PCMLIST.

     MOVE ARTFEC1    TO FECHA.
     MOVE CORR FECHA TO FECHA1.
     MOVE FECHA1           TO "Text"    (11)       OF POW-PCMLIST.

     MOVE ARTIMG1        TO "Text"    (12) OF POW-PCMLIST.
     MOVE ARTOBS         TO "Text"    (13) OF POW-PCMLIST.
     MOVE ARTACT         TO "Text"    (14) OF POW-PCMLIST.

     INVOKE PROGRESO "ProgressStep".
     MOVE WK-IDX TO CONTA.

     IF CONTA = 1000
        OR = 2000
           OR = 3000
              OR = 4000
                 OR = 5000
                    OR = 6000
                       OR = 7000
                          OR = 8000
                             OR = 9000
                                OR = 9999
                                   INVOKE POW-SELF "ThruEvents"
     END-IF.
*
 FIN-CARGAR.
     EXIT.
*
```

Subrutina "CARGAR2"

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  I                        PIC 99    BINARY.
```

```
 01  WTEX                      PIC X(70)        VALUE SPACES.
 01  TAM                       PIC 999   BINARY VALUE 0.
 01  POW-PCMLIST               OBJECT REFERENCE POW-CLISTITEM.
 01  WK-IDX                    PIC S9(9) COMP-5.
 01  CONTA-EDI                 PIC ZZZZZ9.
*
 01  HORA.
     03                        PIC 99.
     03                        PIC 99.
     03                        PIC 99.
     03.
         05  C                 PIC 9.
         05  CE                PIC 9.
*
 01  CONTA                     PIC 9(6) BINARY.
 01  CUATRO                    PIC 9(4) BINARY.
*
 01  FECHA.
     03  AA                    PIC XXXX.
     03  MM                    PIC XX.
     03  DD                    PIC XX.
 01  FECHA1.
     03  DD                    PIC 99.
     03  MM                    PIC 99.
     03  AA                    PIC 9999.
*
 PROCEDURE       DIVISION.
*
     INITIALIZE WS-SQL WTEX CONTASQL CONTA.
*
     MOVE AHIVA15T TO WTEX.
     MOVE SPACES TO AHIVA15T.

     MOVE 11   TO "MousePointer" OF POW-SELF.
     MOVE 1    TO "Visible"      OF PROGRESO.
     MOVE 0    TO "Min"          OF PROGRESO.
     MOVE 2000 TO "Max"          OF PROGRESO.

     STRING "SELECT "
            "ARTCOD, "
                "ARTDES, "
                "ARTFAM, "
                "ARTPRO, "
                "ARTIVA, "
                "ARTPRE1, "
                "ARTPRE2, "
                "ARTPRE3, "
                "ARTOBS, "
                "ARTACT, "
                "ARTIMG1, "
                "ARTFEC, "
                "ARTFEC1 "
            "FROM ARTICULOS WHERE "
            "ARTELI <> 'S' AND "
            DELIMITED BY SIZE INTO WS-SQL
     END-STRING.

     COMPUTE TAM = FUNCTION STORED-CHAR-LENGTH (WS-SQL) + 2.

     COMPUTE I = FUNCTION STORED-CHAR-LENGTH (WTEX).

     STRING "ARTDES LIKE '%"
            WTEX (1:I)
            "%' ORDER BY ARTDES, ARTCOD"
                DELIMITED BY SIZE
                          INTO WS-SQL WITH POINTER TAM
     END-STRING.
```

```
      EXEC SQL
           DECLARE cursor_con3 CURSOR FOR CONSULTA
      END-EXEC.

      EXEC SQL
           PREPARE CONSULTA FROM :WS-SQL
      END-EXEC.

      EXEC SQL
           OPEN cursor_con3
      END-EXEC.

          INVOKE VIEW2 "Clear".

      PERFORM CARGAR THRU FIN-CARGAR
              UNTIL SQLSTATE = "02000".

      EXEC SQL
           CLOSE cursor_con3
      END-EXEC.

      EXEC SQL
           COMMIT
      END-EXEC.

         MOVE CONTASQL TO CONTA-EDI.
         STRING CONTA-EDI
                " Registros. Click en columna para ordenar."
                 DELIMITED BY SIZE INTO WTEX
         END-STRING.
         MOVE WTEX TO "ToolTipText" OF VIEW2.

*
 FIN-LEO.
      IF WK-IDX = 0
           MOVE POW-OFF TO "Enabled" OF VIEW2
      ELSE
           MOVE 1       TO "VISIBLE" OF VIEW2
           MOVE POW-ON  TO "Enabled" OF VIEW2
      END-IF.
      MOVE 0 TO "Visible"      OF PROGRESO.
      MOVE 0 TO "MousePointer" OF POW-SELF.
      MOVE 1 TO "Enabled"      OF BTLISTAR.
          INVOKE CAMPO2 "SetFocus".
*
      EXIT PROGRAM.
*
       CARGAR.

      ACCEPT HORA FROM TIME.

      EXEC SQL
           FETCH cursor_con3
                 INTO
                    :ARTCOD,
                        :ARTDES,
                        :ARTFAM,
                        :ARTPRO,
                        :ARTIVA,
                        :ARTPRE1,
                        :ARTPRE2,
                        :ARTPRE3,
                        :ARTOBS,
                        :ARTACT,
                        :ARTIMG1,
                        :ARTFEC,
                        :ARTFEC1
      END-EXEC.
```

```
      IF SQLSTATE = "02000"
         GO FIN-CARGAR
      END-IF.

      INVOKE VIEW2 "Add" USING 1 1 RETURNING WK-IDX.

      IF WK-IDX > 0 THEN
         MOVE "ListItems"(WK-IDX) OF VIEW2 TO POW-PCMLIST
      END-IF.

      MOVE ARTCOD         TO "Text"    (01) OF POW-PCMLIST.
      MOVE ARTDES         TO "Text"    (02) OF POW-PCMLIST.
      MOVE ARTFAM         TO "Text"    (03) OF POW-PCMLIST.

      MOVE ARTFAM TO FAMCOD.
      READ FAMILIAS NO LOCK
           INVALID KEY
                   MOVE " No existe en fichero ..." TO "Text"    (04)    OF POW-PCMLIST
           NOT INVALID KEY
                   MOVE FAMDES                      TO "Text"    (04)    OF POW-PCMLIST
      END-READ.

      MOVE ARTPRO         TO "Numeric" (05) OF POW-PCMLIST.
      MOVE ARTIVA         TO "Numeric" (06) OF POW-PCMLIST.
      MOVE ARTPRE1        TO "Numeric" (07) OF POW-PCMLIST.
      MOVE ARTPRE2        TO "Numeric" (08) OF POW-PCMLIST.
      MOVE ARTPRE3        TO "Numeric" (09) OF POW-PCMLIST.

      MOVE ARTFEC     TO FECHA.
      MOVE CORR FECHA TO FECHA1.
      MOVE FECHA1           TO "Text"    (10)       OF POW-PCMLIST.

      MOVE ARTFEC1    TO FECHA.
      MOVE CORR FECHA TO FECHA1.
      MOVE FECHA1           TO "Text"    (11)       OF POW-PCMLIST.

      MOVE ARTIMG1        TO "Text"    (12) OF POW-PCMLIST.
      MOVE ARTOBS         TO "Text"    (13) OF POW-PCMLIST.
      MOVE ARTACT         TO "Text"    (14) OF POW-PCMLIST.

      ADD 1 TO CONTASQL.

      COMPUTE CONTA = CONTA + CE.
      IF CONTA > 2000
         MOVE 1 TO CONTA
      END-IF.
      INVOKE PROGRESO "ProgressStep".
*
      MOVE CONTASQL TO CUATRO.

      IF CUATRO = 1000
         OR = 2000
            OR = 3000
               OR = 4000
                  OR = 5000
                     OR = 6000
                        OR = 7000
                           OR = 8000
                              OR = 9000
                                 OR = 9999
                                    INVOKE POW-SELF "ThruEvents"
      END-IF.
*
 FIN-CARGAR.
      EXIT.
*
```

Subrutina "CARGAR-N"

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  I                        PIC 99  COMP-5 VALUE 0.
 01  WTEX                     PIC X(60)      VALUE SPACES.
 01  TAM                      PIC 999        VALUE 0.
 01  WSSINDEC                 PIC S99999     SIGN IS LEADING SEPARATE CHARACTER.
 01  WSFEC              PIC S9(8)      SIGN IS LEADING SEPARATE CHARACTER.
 01  WSCONDEC                 PIC S9(7)V999  SIGN IS LEADING SEPARATE CHARACTER.
 01  WSDECIMAL                PIC 9.            *> CANTIDAD DE DECIMALES
 01  WSSINDEC-EDI             PIC ZZZZZ9.
 01  WSCONDEC-EDI             PIC -------9,999.
 01  WSFEC-EDI                PIC ZZZZZZZ9.
 01  WSOCHO.
     03  WS8                  PIC X OCCURS 12.
 01  WSSEIS.
     03  WS6                  PIC X OCCURS 6.
 01  WSFECHA.
     03  WSF                  PIC X OCCURS 8.
*
 01  POW-PCMLIST              OBJECT REFERENCE POW-CLISTITEM.
 01  WK-IDX                   PIC S9(9) COMP-5.
 01  CONTA-EDI                PIC ZZZZZ9.
*
 01  HORA.
     03                       PIC 99.
     03                       PIC 99.
     03                       PIC 99.
     03.
         05  C                PIC 9.
         05  CE               PIC 9.
*
 01  CONTA                    PIC 9(6) BINARY.
 01  CUATRO                   PIC 9(4) BINARY.
*
 01  FECHA.
     03  AA                   PIC XXXX.
     03  MM                   PIC XX.
     03  DD                   PIC XX.
 01  FECHA1.
     03  DD                   PIC 99.
     03  MM                   PIC 99.
     03  AA                   PIC 9999.
*
 PROCEDURE       DIVISION.
*
     INITIALIZE WS-SQL WTEX CONTASQL CONTA.
*
     MOVE 11   TO "MousePointer" OF POW-SELF.
     MOVE 1    TO "Visible"      OF PROGRESO.
     MOVE 0    TO "Min"          OF PROGRESO.
     MOVE 2000 TO "Max"          OF PROGRESO.
*
     STRING "SELECT "
            "ARTCOD, "
                "ARTDES, "
                "ARTFAM, "
                "ARTPRO, "
                "ARTIVA, "
                "ARTPRE1, "
                "ARTPRE2, "
                "ARTPRE3, "
                "ARTOBS, "
                "ARTACT, "
```

```
                "ARTIMG1, "
                "ARTFEC, "
                "ARTFEC1 "
            "FROM ARTICULOS WHERE "
            DELIMITED BY SIZE INTO WS-SQL
      END-STRING.

      COMPUTE TAM = FUNCTION STORED-CHAR-LENGTH (WS-SQL) + 2.

      STRING "ARTELI <> 'S' AND "
            DELIMITED BY SIZE INTO WS-SQL WITH POINTER TAM
      END-STRING.

      EVALUATE "ListIndex" OF COMBO2

               WHEN 4                *> PROVEEDOR
                    MOVE "Text" OF CAMPO1 TO WSSINDEC
                    MOVE "Text" OF CAMPO1 TO WSSINDEC-EDI
                    INITIALIZE WSSEIS
                    MOVE WSSINDEC-EDI TO WSSEIS

                    PERFORM VARYING I FROM 1 BY 1
                            UNTIL WS6 (I) IS NUMERIC, OR I > 6
                                  CONTINUE
                    END-PERFORM

                    IF I > 6
                       MOVE 1 TO I
                    END-IF

                    STRING "ARTFAM LIKE '%"
                           WSSEIS (I:(7 - I))
                           "%' ORDER BY ARTPRO ASC"
                           DELIMITED BY SIZE INTO WS-SQL
                                     WITH POINTER TAM
                    END-STRING

               WHEN 5           *> IVA
                    MOVE "Text" OF CAMPO1 TO WSCONDEC
                    MOVE "Text" OF CAMPO1 TO WSCONDEC-EDI
                    INITIALIZE WSOCHO
                    MOVE WSCONDEC-EDI TO WSOCHO

                    PERFORM VARYING I FROM 1 BY 1
                            UNTIL WS8 (I) IS NUMERIC, OR I > 12, OR WS8 (I) = "-"
                                  CONTINUE
                    END-PERFORM

                    IF I > 13
                       MOVE 1 TO I
                    END-IF

                    INSPECT WSOCHO REPLACING ALL "," BY "."  *> Porque SQL utiliza
notación Inglesa para los decimales

                    STRING "ARTIVA LIKE '%"
                           WSOCHO (I:(13 - I))
                           "%' ORDER BY ARTIVA ASC"
                           DELIMITED BY SIZE INTO WS-SQL
                                     WITH POINTER TAM
                    END-STRING

               WHEN 6           *> PRECIO 1
                    MOVE "Text" OF CAMPO1 TO WSCONDEC
                    MOVE "Text" OF CAMPO1 TO WSCONDEC-EDI
                    INITIALIZE WSOCHO
                    MOVE WSCONDEC-EDI TO WSOCHO
```

```
                PERFORM VARYING I FROM 1 BY 1
                        UNTIL WS8 (I) IS NUMERIC, OR I > 12, OR WS8 (I) = "-"
                             CONTINUE
                END-PERFORM

                IF I > 13
                   MOVE 1 TO I
                END-IF

                INSPECT WSOCHO REPLACING ALL "," BY "."  *> Porque SQL utiliza
notación Inglesa para los decimales

                STRING "ARTPRE1 LIKE '%"    *>        6.000
                       WSOCHO (I:(13 - I))  *> 12345678,901
                       "%' ORDER BY ARTPRE1 ASC"
                       DELIMITED BY SIZE INTO WS-SQL
                                 WITH POINTER TAM
                END-STRING

           WHEN 7          *> PRECIO 2
                MOVE "Text" OF CAMPO1 TO WSCONDEC
                MOVE "Text" OF CAMPO1 TO WSCONDEC-EDI
                INITIALIZE WSOCHO
                MOVE WSCONDEC-EDI TO WSOCHO

                PERFORM VARYING I FROM 1 BY 1
                        UNTIL WS8 (I) IS NUMERIC, OR I > 12, OR WS8 (I) = "-"
                             CONTINUE
                END-PERFORM

                IF I > 13
                   MOVE 1 TO I
                END-IF

                INSPECT WSOCHO REPLACING ALL "," BY "."  *> Porque SQL utiliza
notación Inglesa para los decimales

                STRING "ARTPRE2 LIKE '%"
                       WSOCHO (I:(13 - I))
                       "%' ORDER BY ARTPRE2 ASC"
                       DELIMITED BY SIZE INTO WS-SQL
                                 WITH POINTER TAM
                END-STRING

           WHEN 8          *> PRECIO 3
                MOVE "Text" OF CAMPO1 TO WSCONDEC
                MOVE "Text" OF CAMPO1 TO WSCONDEC-EDI
                INITIALIZE WSOCHO
                MOVE WSCONDEC-EDI TO WSOCHO

                PERFORM VARYING I FROM 1 BY 1
                        UNTIL WS8 (I) IS NUMERIC, OR I > 12, OR WS8 (I) = "-"
                             CONTINUE
                END-PERFORM

                IF I > 13
                   MOVE 1 TO I
                END-IF

                INSPECT WSOCHO REPLACING ALL "," BY "."  *> Porque SQL utiliza
notación Inglesa para los decimales

                STRING "ARTPRE3 LIKE '%"
                       WSOCHO (I:(13 - I))
                       "%' ORDER BY ARTPRE3 ASC"
```

```
                    DELIMITED BY SIZE INTO WS-SQL
                              WITH POINTER TAM
               END-STRING

          WHEN 9          *> FECHA ALTA
               MOVE "Text" OF CAMPO1 TO WSFEC
               MOVE "Text" OF CAMPO1 TO WSFEC-EDI
               INITIALIZE WSFECHA
               MOVE WSFEC-EDI TO WSFECHA

               PERFORM VARYING I FROM 1 BY 1
                       UNTIL WSF (I) IS NUMERIC, OR I > 8
                             CONTINUE
               END-PERFORM

               IF I > 8
                  MOVE 1 TO I
               END-IF

               STRING "ARTFEC LIKE '%"
                      WSFECHA (I:(6 - I))
                      "%' ORDER BY ARTFEC ASC"
                      DELIMITED BY SIZE INTO WS-SQL
                                WITH POINTER TAM
               END-STRING

          WHEN 10         *> FECHA ÚLTIMA MODIFICACIÓN
               MOVE "Text" OF CAMPO1 TO WSFEC
               MOVE "Text" OF CAMPO1 TO WSFEC-EDI
               INITIALIZE WSFECHA
               MOVE WSFEC-EDI TO WSFECHA

               PERFORM VARYING I FROM 1 BY 1
                       UNTIL WSF (I) IS NUMERIC, OR I > 8
                             CONTINUE
               END-PERFORM

               IF I > 8
                  MOVE 1 TO I
               END-IF

               STRING "ARTFEC1 LIKE '%"
                      WSFECHA (I:(6 - I))
                      "%' ORDER BY ARTFEC1 ASC"
                      DELIMITED BY SIZE INTO WS-SQL
                                WITH POINTER TAM
               END-STRING


          WHEN OTHER
               INVOKE CAMPO1 "SetFocus"
               EXIT PROGRAM
END-EVALUATE.

EXEC SQL
     DECLARE cursor_con2 CURSOR FOR CONSULTA
END-EXEC.

EXEC SQL
     PREPARE CONSULTA FROM :WS-SQL
END-EXEC.

EXEC SQL
     OPEN cursor_con2
END-EXEC.

    INVOKE VIEW2 "Clear".
```

```
      PERFORM CARGAR THRU FIN-CARGAR
              UNTIL SQLSTATE = "02000".

      EXEC SQL
           COMMIT
      END-EXEC.

         MOVE CONTASQL TO CONTA-EDI.
         STRING CONTA-EDI
                "Registros. Click en columna para ordenar."
                 DELIMITED BY SIZE INTO WTEX
         END-STRING.
         MOVE WTEX TO "ToolTipText" OF VIEW2.

*
 FIN-LEO.
      IF WK-IDX = 0
           MOVE POW-OFF TO "Enabled" OF VIEW2
      ELSE
           MOVE 1       TO "VISIBLE" OF VIEW2
           MOVE POW-ON  TO "Enabled" OF VIEW2
      END-IF.
      MOVE 0 TO "Visible"      OF PROGRESO.
      MOVE 0 TO "MousePointer" OF POW-SELF.
      MOVE 1 TO "Enabled"      OF BTLISTAR.
          INVOKE CAMPO1 "SetFocus".
*
      EXIT PROGRAM.
*
       CARGAR.

      ACCEPT HORA FROM TIME.

      EXEC SQL
           FETCH cursor_con2
                 INTO
                    :ARTCOD,
                        :ARTDES,
                        :ARTFAM,
                        :ARTPRO,
                        :ARTIVA,
                        :ARTPRE1,
                        :ARTPRE2,
                        :ARTPRE3,
                        :ARTOBS,
                        :ARTACT,
                        :ARTIMG1,
                        :ARTFEC,
                        :ARTFEC1
      END-EXEC.

      IF SQLSTATE = "02000"
         GO FIN-CARGAR
      END-IF.

      INVOKE VIEW2 "Add" USING 1 1 RETURNING WK-IDX.

      IF WK-IDX > 0 THEN
         MOVE "ListItems"(WK-IDX) OF VIEW2 TO POW-PCMLIST
      END-IF.

      MOVE ARTCOD         TO "Text"     (01) OF POW-PCMLIST.
      MOVE ARTDES         TO "Text"     (02) OF POW-PCMLIST.
      MOVE ARTFAM         TO "Text"     (03) OF POW-PCMLIST.

      MOVE ARTFAM TO FAMCOD.
      READ FAMILIAS NO LOCK
```

```
          INVALID KEY
                  MOVE " No existe en fichero ..." TO "Text"    (04)    OF POW-PCMLIST
          NOT INVALID KEY
                  MOVE FAMDES                      TO "Text"    (04)    OF POW-PCMLIST
      END-READ.

      MOVE ARTPRO         TO "Numeric" (05) OF POW-PCMLIST.
      MOVE ARTIVA         TO "Numeric" (06) OF POW-PCMLIST.
      MOVE ARTPRE1        TO "Numeric" (07) OF POW-PCMLIST.
      MOVE ARTPRE2        TO "Numeric" (08) OF POW-PCMLIST.
      MOVE ARTPRE3        TO "Numeric" (09) OF POW-PCMLIST.

      MOVE ARTFEC     TO FECHA.
      MOVE CORR FECHA TO FECHA1.
      MOVE FECHA1           TO "Text"    (10)       OF POW-PCMLIST.

      MOVE ARTFEC1    TO FECHA.
      MOVE CORR FECHA TO FECHA1.
      MOVE FECHA1           TO "Text"    (11)       OF POW-PCMLIST.

      MOVE ARTIMG1        TO "Text"    (12) OF POW-PCMLIST.
      MOVE ARTOBS         TO "Text"    (13) OF POW-PCMLIST.
      MOVE ARTACT         TO "Text"    (14) OF POW-PCMLIST.

      ADD 1 TO CONTASQL.

      COMPUTE CONTA = CONTA + CE.
      IF CONTA > 2000
         MOVE 1 TO CONTA
      END-IF.
      INVOKE PROGRESO "ProgressStep".

      MOVE CONTASQL TO CUATRO.

      IF CUATRO = 1000
         OR = 2000
            OR = 3000
               OR = 4000
                  OR = 5000
                     OR = 6000
                        OR = 7000
                           OR = 8000
                              OR = 9000
                                 OR = 9999
                                    INVOKE POW-SELF "ThruEvents"
      END-IF.
*
 FIN-CARGAR.
      EXIT.
*
```

Subrutina “CARGAR-N”

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01   I                        PIC 99    BINARY.
 01   WTEX                     PIC X(70) VALUE SPACES.
 01   TAM                      PIC 999   BINARY VALUE 0.
 01   POW-PCMLIST              OBJECT REFERENCE POW-CLISTITEM.
 01   WK-IDX                   PIC S9(9) COMP-5.
 01   CONTA-EDI                PIC ZZZZZ9.
*
 01   HORA.
      03                       PIC 99.
      03                       PIC 99.
```

```
     03                        PIC 99.
     03.
         05  C                 PIC 9.
         05  CE                PIC 9.
*
 01  CONTA              PIC 9(6) BINARY.
 01  CUATRO             PIC 9(4) BINARY.
*
 01  FECHA.
     03  AA             PIC XXXX.
     03  MM             PIC XX.
     03  DD             PIC XX.
 01  FECHA1.
     03  DD             PIC 99.
     03  MM             PIC 99.
     03  AA             PIC 9999.
*
 PROCEDURE       DIVISION.
*
     INITIALIZE WS-SQL WTEX CONTASQL CONTA.
*
     IF "Text" OF CAMPO2 = SPACES
        INVOKE CAMPO2 "SetFocus"
        EXIT PROGRAM
     ELSE
        MOVE "Text" OF CAMPO2 TO WTEX
     END-IF.

     MOVE 11   TO "MousePointer" OF POW-SELF.
     MOVE 1    TO "Visible"      OF PROGRESO.
     MOVE 0    TO "Min"          OF PROGRESO.
     MOVE 2000 TO "Max"          OF PROGRESO.

     STRING "SELECT "
            "ARTCOD, "
                "ARTDES, "
                "ARTFAM, "
                "ARTPRO, "
                "ARTIVA, "
                "ARTPRE1, "
                "ARTPRE2, "
                "ARTPRE3, "
                "ARTOBS, "
                "ARTACT, "
                "ARTIMG1, "
                "ARTFEC, "
                "ARTFEC1 "
            "FROM ARTICULOS WHERE "
            DELIMITED BY SIZE INTO WS-SQL
     END-STRING.

     COMPUTE TAM = FUNCTION STORED-CHAR-LENGTH (WS-SQL) + 2.

     STRING "ARTELI <> 'S' AND "
            DELIMITED BY SIZE INTO WS-SQL WITH POINTER TAM
     END-STRING.

     COMPUTE I = FUNCTION STORED-CHAR-LENGTH (WTEX).

     EVALUATE "ListIndex" OF COMBO2

              WHEN 1
                   STRING "ARTCOD LIKE '%"
                          WTEX (1:I)
                          "%' ORDER BY ARTCOD ASC"
                              DELIMITED BY SIZE
                              INTO WS-SQL WITH POINTER TAM
                   END-STRING
```

```
           WHEN 2
                STRING "ARTDES LIKE '%"
                       WTEX (1:I)
                       "%' ORDER BY ARTDES ASC"
                           DELIMITED BY SIZE
                           INTO WS-SQL WITH POINTER TAM
                END-STRING

           WHEN 3
                STRING "ARTFAM LIKE '%"
                       WTEX (1:I)
                       "%' ORDER BY ARTFAM ASC"
                           DELIMITED BY SIZE
                           INTO WS-SQL WITH POINTER TAM
                END-STRING

           WHEN 11
                STRING "ARTIMG1 LIKE '%"
                       WTEX (1:I)
                       "%' ORDER BY ARTIMG1 ASC"
                           DELIMITED BY SIZE
                           INTO WS-SQL WITH POINTER TAM
                END-STRING

           WHEN 12
                STRING "ARTOBS LIKE '%"
                       WTEX (1:I)
                       "%' ORDER BY ARTOBS ASC"
                           DELIMITED BY SIZE
                           INTO WS-SQL WITH POINTER TAM
                END-STRING

END-EVALUATE.

EXEC SQL
     DECLARE cursor_con1 CURSOR FOR CONSULTA
END-EXEC.

EXEC SQL
     PREPARE CONSULTA FROM :WS-SQL
END-EXEC.

EXEC SQL
     OPEN cursor_con1
END-EXEC.

    INVOKE VIEW2 "Clear".

PERFORM CARGAR THRU FIN-CARGAR
        UNTIL SQLSTATE = "02000".

EXEC SQL
     CLOSE cursor_con1
END-EXEC.

EXEC SQL
     COMMIT
END-EXEC.

   MOVE CONTASQL TO CONTA-EDI.
   STRING CONTA-EDI
          " Registros. Click en columna para ordenar."
           DELIMITED BY SIZE INTO WTEX
   END-STRING.
   MOVE WTEX TO "ToolTipText" OF VIEW2.
```

```
*
 FIN-LEO.
     IF WK-IDX = 0
          MOVE POW-OFF TO "Enabled" OF VIEW2
     ELSE
          MOVE 1       TO "VISIBLE" OF VIEW2
          MOVE POW-ON  TO "Enabled" OF VIEW2
     END-IF.
     MOVE 0 TO "Visible"      OF PROGRESO.
     MOVE 0 TO "MousePointer" OF POW-SELF.
     MOVE 1 TO "Enabled"      OF BTLISTAR.
         INVOKE CAMPO2 "SetFocus".
*
     EXIT PROGRAM.
*
      CARGAR.

     ACCEPT HORA FROM TIME.

     EXEC SQL
          FETCH cursor_con1
                INTO
                   :ARTCOD,
                       :ARTDES,
                       :ARTFAM,
                       :ARTPRO,
                       :ARTIVA,
                       :ARTPRE1,
                       :ARTPRE2,
                       :ARTPRE3,
                       :ARTOBS,
                       :ARTACT,
                       :ARTIMG1,
                       :ARTFEC,
                       :ARTFEC1
     END-EXEC.

     IF SQLSTATE = "02000"
        GO FIN-CARGAR
     END-IF.

     INVOKE VIEW2 "Add" USING 1 1 RETURNING WK-IDX.

     IF WK-IDX > 0 THEN
        MOVE "ListItems"(WK-IDX) OF VIEW2 TO POW-PCMLIST
     END-IF.

     MOVE ARTCOD         TO "Text"    (01) OF POW-PCMLIST.
     MOVE ARTDES         TO "Text"    (02) OF POW-PCMLIST.
     MOVE ARTFAM         TO "Text"    (03) OF POW-PCMLIST.

     MOVE ARTFAM TO FAMCOD.
     READ FAMILIAS NO LOCK
          INVALID KEY
                  MOVE " No existe en fichero ..." TO "Text"    (04)    OF POW-PCMLIST
          NOT INVALID KEY
                  MOVE FAMDES                      TO "Text"    (04)    OF POW-PCMLIST
     END-READ.

     MOVE ARTPRO         TO "Numeric" (05) OF POW-PCMLIST.
     MOVE ARTIVA         TO "Numeric" (06) OF POW-PCMLIST.
     MOVE ARTPRE1        TO "Numeric" (07) OF POW-PCMLIST.
     MOVE ARTPRE2        TO "Numeric" (08) OF POW-PCMLIST.
     MOVE ARTPRE3        TO "Numeric" (09) OF POW-PCMLIST.

     MOVE ARTFEC     TO FECHA.
     MOVE CORR FECHA TO FECHA1.
     MOVE FECHA1           TO "Text"    (10)       OF POW-PCMLIST.
```

```
      MOVE ARTFEC1    TO FECHA.
      MOVE CORR FECHA TO FECHA1.
      MOVE FECHA1          TO "Text"    (11)       OF POW-PCMLIST.

      MOVE ARTIMG1       TO "Text"    (12) OF POW-PCMLIST.
      MOVE ARTOBS        TO "Text"    (13) OF POW-PCMLIST.
      MOVE ARTACT        TO "Text"    (14) OF POW-PCMLIST.

      ADD 1 TO CONTASQL.

      COMPUTE CONTA = CONTA + CE.
      IF CONTA > 2000
         MOVE 1 TO CONTA
      END-IF.
      INVOKE PROGRESO "ProgressStep".

      MOVE CONTASQL TO CUATRO.

      IF CUATRO = 1000
         OR = 2000
            OR = 3000
               OR = 4000
                  OR = 5000
                     OR = 6000
                        OR = 7000
                           OR = 8000
                              OR = 9000
                                 OR = 9999
                                    INVOKE POW-SELF "ThruEvents"
      END-IF.
*
 FIN-CARGAR.
      EXIT.
```

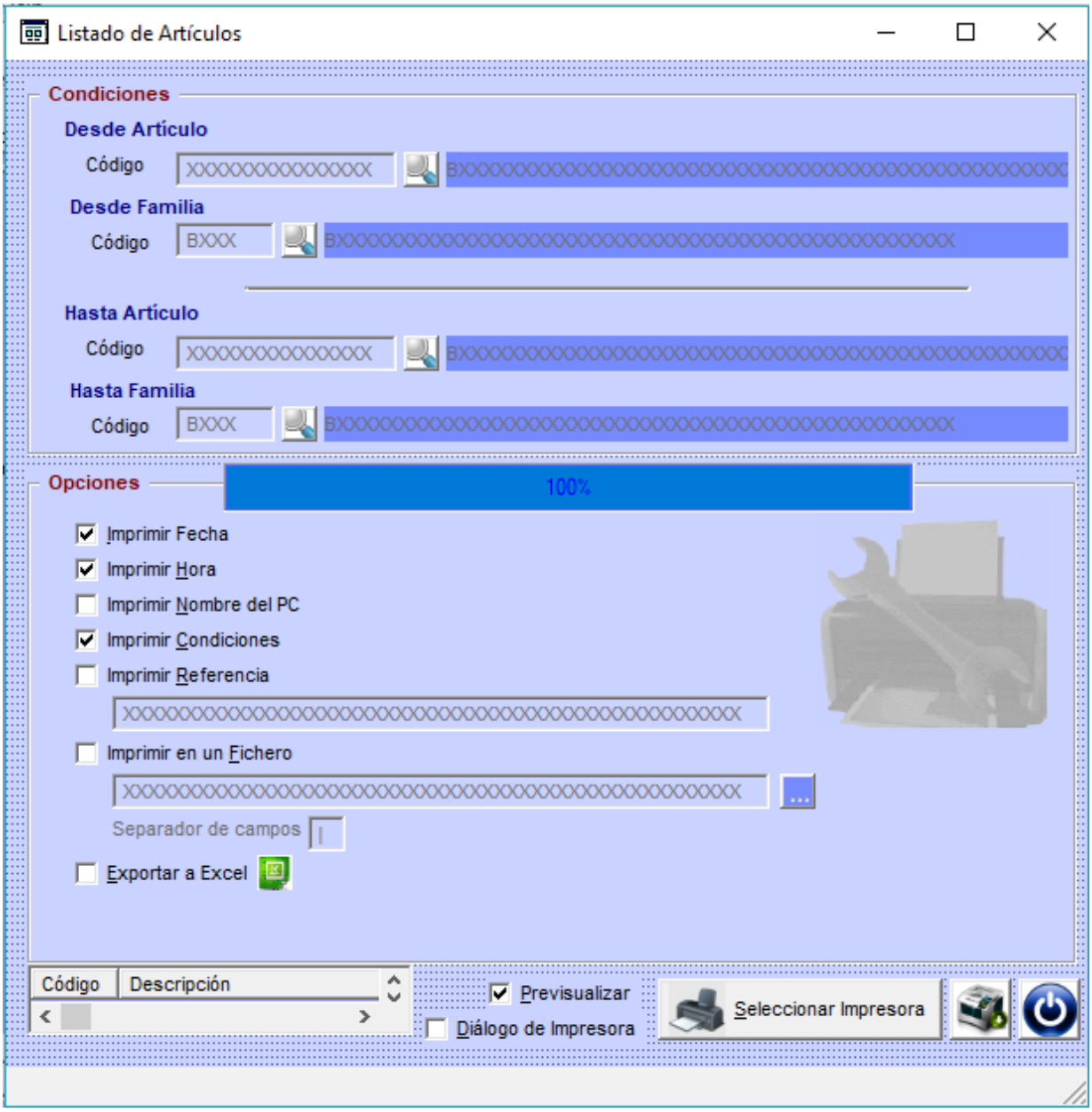
Subrutina "PARA-SALIR"

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.
    INVOKE POW-SELF "CloseForm".
```

El Programa "*LISTADO*"

Para terminar, sólo nos queda crear el programa para el listado. En él podremos elegir entre rango de artículos y a su vez, rango de familias, así como imprimirlo por pantalla, exportarlo a Excel o a un fichero de texto plano, también tenemos la opción de seleccionar la impresora por la que emitir el listado.

Realizamos los mismos pasos que hicimos para crear el nuevo formulario de consulta, a éste lo llamaremos **F-LIS-ART**, ajustamos las propiedades, creamos todos los elementos del form y a mi me ha quedado así "de majo"



Por cierto, para poder exportar a Excel, lo haremos cargando los datos en un *listView*, que nunca será visible para el usuario final y que será el que exportemos.

Programación del Form (Listado)

Click con el botón derecho en el form y vamos a definir los datos comunes a todo el form

ENVIRONMENT DIVISION → SPECIAL-NAMES:

```
   DECIMAL-POINT IS COMMA.
```

ENVIRONMENT DIVISION → REPOSITORY:

```
     CLASS OLE       AS "*OLE"
     CLASS OLE-EX    AS "*OLE-EXCEPTION"
     CLASS COM-ARRAY AS "*COM-ARRAY".
```

ENVIRONMENT DIVISION → FILE CONTROL:

```
    SELECT OPTIONAL FAMILIAS
             ASSIGN TO "C:\Articulos\Datos\FAMILIAS.FIC"
             ORGANIZATION IS INDEXED
             ACCESS MODE  IS DYNAMIC
             RECORD KEY   IS FAMCOD
             FILE STATUS  IS STFAM
             LOCK IS AUTOMATIC.
*
     SELECT OPTIONAL
             FICHERO ASSIGN TO FILE-TEXT
             ORGANIZATION IS LINE SEQUENTIAL
             FILE STATUS IS STFIC.
```

DATA DIVISION → FILE SECTION:

```
 FD  FAMILIAS GLOBAL EXTERNAL
     LABEL RECORD IS STANDARD
     DATA RECORD IS REG-FAM.
 01  REG-FAM.
     03  FAMCOD               PIC XXX.
     03  FAMDES               PIC X(30).
*
 FD  FICHERO GLOBAL
     LABEL RECORD IS STANDARD.
 01  FICTEXT                  PIC X(1912).
```

DATA DIVISION → WORKING-STORAGE SECTION:

```
 01  ESTILO                   PIC S9(4) COMP-5   GLOBAL.
 01  RETORNO                  PIC S9(4) COMP-5   GLOBAL.
 01  WS-ATENCION              PIC X(17)          GLOBAL VALUE " ¡ Atención ... !".
 01  SALTO                    PIC X  VALUE X"0A" GLOBAL.
 01  AHIVA3T                  PIC XXX            GLOBAL EXTERNAL.
 01  AHIVA15T                 PIC X(15)          GLOBAL EXTERNAL.
*
* ----- * -----
*
 01  STFIC                    PIC XX           GLOBAL.
 01  STFAM                    PIC XX           GLOBAL.
 01  FILE-TEXT                PIC X(255)       GLOBAL.
 01  WMAQ                     PIC X(32)        GLOBAL VALUE SPACES.
 01  CONTA                    PIC 9999         GLOBAL.
*
* --- * DECLARACIÓN DE LAS VARIABLES PARA SQL
```

```
*
                EXEC SQL
                     BEGIN DECLARE SECTION
                END-EXEC.
*
 01  REG-ART      GLOBAL.
     03  ARTCOD                PIC X(15).
     03  ARTDES                PIC X(60).
     03  ARTFAM                PIC XXX.
     03  ARTPRO                PIC S9(6).
     03  ARTIVA                PIC S999V99.
     03  ARTPRE1               PIC S9(7)V999.
     03  ARTPRE2               PIC S9(7)V999.
     03  ARTPRE3               PIC S9(7)V999.
     03  ARTOBS                PIC X(200).
     03  ARTACT                PIC X.
     03  ARTIMG1               PIC X(255).
     03  ARTIMG2               PIC X(255).
     03  ARTIMG3               PIC X(255).
     03  ARTIMG4               PIC X(255).
     03  ARTIMG5               PIC X(255).
     03  ARTIMG6               PIC X(255).
     03  ARTFEC                PIC X(8).
     03  ARTFEC1               PIC X(8).
     03  ARTELI                PIC X.      *>  Para  saber  si  un  artículo  ha  sido
eliminado (S/N).
*
* --- VARIABLES de LECTURA
*
  01  WNUM1 IS GLOBAL.
      03  WCOD10               PIC X(15).
  01  WNUM2 IS GLOBAL.
      03  WCOD20               PIC X(15).
  01  WNUM3 IS GLOBAL.
      03  WCOD30               PIC X(3).
  01  WNUM4 IS GLOBAL.
      03  WCOD40               PIC X(3).
      *>
      *> --- VARIABLES
      *>
 01  SQLSTATE                  PIC XXXXX   IS GLOBAL.
 01  SQLCODE                   PIC S9(9) COMP-5   GLOBAL.
 01  SQLMSG                    PIC X(600)     GLOBAL.
 01  CONTASQL                  PIC S9(9)   IS GLOBAL.  *>   CONTADOR   DE   NÚMERO   DE
REGISTROS
 01  WS-SQL                    PIC X(350)  IS GLOBAL.
*
           EXEC SQL
                END DECLARE SECTION
           END-EXEC.
*
*
  01  WSPRINTER                PIC X(64) GLOBAL EXTERNAL.
*
  01  WTEXTOS     IS GLOBAL.
      03  WTEXT                PIC X(100).
  01  FECHA IS GLOBAL.
      03  AA                   PIC XXXX.
      03  MM                   PIC XX.
      03  DD                   PIC XX.
  01  FECHA1      IS GLOBAL.
      03  DD                   PIC 99.
      03    MM                  PIC 99.
      03  AA                   PIC 9999.
  01  FECHA-EDI   IS GLOBAL.
      03  DD                   PIC 99.
      03                       PIC X VALUE "-".
      03  MM                   PIC 99.
      03                       PIC X VALUE "-".
```

```
    03  AA                  PIC 9999.
01  HORA  IS GLOBAL.
    03  HH                  PIC 99.
    03  MM                  PIC 99.
    03  SS                  PIC 99.
01  HORA-EDI    IS GLOBAL.
    03  HH                  PIC 99.
    03                      PIC X VALUE ":".
    03  MM                  PIC 99.
    03                      PIC X VALUE "'".
    03  SS                  PIC 99.
```

Eventos del Form

Evento *Opened* del form:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
*
* ---- VARIABLE PARA EL NOMBRE DEL ORDENADOR
*
 01  RESULT                  PIC S9(9) COMP-5.
 01  COMPNAME                PIC X(32) VALUE SPACES.
 01  LENCOMPNAME             PIC S9(9) COMP-5.
*
 PROCEDURE       DIVISION.
 INICIO.
*
* ----- SACAMOS EL NOMBRE DEL PC y lo ALMACENAMOS en la VARIABLE WMAQ
*
            MOVE 32 TO LENCOMPNAME.

            CALL "GetComputerNameA" WITH STDCALL LINKAGE
                 USING
                       BY REFERENCE COMPNAME
                   BY CONTENT     LENCOMPNAME
                 RETURNING RESULT
            END-CALL.

      MOVE COMPNAME TO WMAQ.
      INSPECT WMAQ
              REPLACING ALL LOW-VALUES BY SPACES.

            CANCEL "GetComputerNameA".
*
*
     INVOKE POW-SELF "ShowForm" USING POW-SWSHOWNORMAL
     CALL "VACIAR".
     INVOKE CAMPO1 "SetFocus".
```

Eventos de los controles del Form

Evento *Return* de campo1:

```
ENVIRONMENT     DIVISION.
DATA            DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE       DIVISION.

    IF "Text" OF CAMPO1 = SPACES
       MOVE " Desde el primero ..." TO "Caption" OF INFORMACION1
       MOVE SPACES TO "Text" OF CAMPO1
       INVOKE CAMPO2 "SetFocus"
       EXIT PROGRAM
    END-IF.

    MOVE "Text" OF CAMPO1 TO ARTCOD.

    EXEC SQL
         SELECT ARTDES FROM ARTICULOS
                WHERE ARTCOD = :ARTCOD
                      INTO :ARTDES
    END-EXEC.

    IF SQLSTATE NOT = "00000"
       MOVE " No Existe ..." TO "Caption" OF INFORMACION1
    ELSE
       MOVE ARTDES TO "Caption" OF INFORMACION1
    END-IF.

    INVOKE CAMPO2 "SetFocus".
    EXIT PROGRAM.
```

Evento *Click* de botón BTCOD1:

```
ENVIRONMENT     DIVISION.
DATA            DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE       DIVISION.

    IF POW-TEXT OF CAMPO1 = SPACES
       MOVE SPACES TO AHIVA15T
    ELSE
       MOVE POW-TEXT OF CAMPO1 TO AHIVA15T
    END-IF.

    INVOKE POW-SELF "CallForm" USING "F-CON-ARTICULOS".

    IF AHIVA15T = SPACES
       INVOKE BTCOD1 "SetFocus"
       EXIT PROGRAM
    END-IF.

    MOVE AHIVA15T TO POW-TEXT OF CAMPO1.
    MOVE AHIVA15T TO ARTCOD.

    EXEC SQL
         SELECT ARTDES FROM ARTICULOS
                WHERE ARTCOD = :ARTCOD
                      INTO :ARTDES
    END-EXEC.
```

```
    IF SQLSTATE NOT = "00000"
       MOVE " No Existe ..." TO "Caption" OF INFORMACION1
    ELSE
       MOVE ARTDES TO "Caption" OF INFORMACION1
    END-IF.

    INVOKE CAMPO2 "SetFocus".
    EXIT PROGRAM.
```

Evento *Return* de CAMPO2:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.

    IF POW-TEXT OF CAMPO2 = SPACES
       MOVE POW-COLOR-BLACK TO "ForeColor" OF INFORMACION2
       MOVE " Desde la primera ..." TO "Caption" OF INFORMACION2
       MOVE SPACES TO POW-TEXT OF CAMPO2
       INVOKE CAMPO3 "SetFocus"
       EXIT PROGRAM
    END-IF.

    MOVE POW-COLOR-BLACK TO "ForeColor" OF INFORMACION2.

    MOVE POW-TEXT OF CAMPO2 TO FAMCOD.

    READ FAMILIAS NO LOCK
         INVALID KEY
                 MOVE " No Existe ..." TO "Caption" OF INFORMACION2
         NOT INVALID KEY
             MOVE FAMDES TO "Caption" OF INFORMACION2
    END-READ.

    INVOKE CAMPO3 "SetFocus".
    EXIT PROGRAM.
```

Evento *Click* de botón BTCOD2:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.

    MOVE SPACES TO AHIVA3T.
    INVOKE POW-SELF "CallForm" USING "F-CON-FAMILIAS" "M-FAMILIAS".

    IF AHIVA3T = SPACES
       INVOKE BTCOD2 "SetFocus"
       EXIT PROGRAM
    END-IF.

    MOVE AHIVA3T TO "Text" OF CAMPO2.
    MOVE AHIVA3T TO FAMCOD.

    READ FAMILIAS NO LOCK
         INVALID KEY
                 MOVE " No Existe ..." TO "Caption" OF INFORMACION2
         NOT INVALID KEY
             MOVE FAMDES TO "Caption" OF INFORMACION2
    END-READ.
```

```
    INVOKE CAMPO3 "SetFocus".
    EXIT PROGRAM.
```

Evento *Return* de CAMPO3:

```
ENVIRONMENT     DIVISION.
DATA            DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE       DIVISION.

    IF POW-TEXT OF CAMPO3 = SPACES
       MOVE POW-COLOR-BLACK TO "ForeColor" OF INFORMACION3
       MOVE " ... Hasta el último" TO "Caption" OF INFORMACION3
       MOVE SPACES TO POW-TEXT OF CAMPO3
       MOVE HIGH-VALUES TO WCOD20
       INVOKE CAMPO4 "SetFocus"
       EXIT PROGRAM
    END-IF.

    IF POW-TEXT OF CAMPO1 > POW-TEXT OF CAMPO3
       MOVE " El primero no puede ser mayor que el segundo ..."
            TO "Caption" OF INFORMACION3
       MOVE POW-COLOR-DARKRED TO "ForeColor" OF INFORMACION3
       INVOKE CAMPO3 "SetFocus"
       EXIT PROGRAM
    END-IF.

    MOVE POW-COLOR-BLACK TO "ForeColor" OF INFORMACION3.

    MOVE POW-TEXT OF CAMPO3 TO ARTCOD.

    EXEC SQL
         SELECT ARTDES FROM ARTICULOS
                WHERE ARTCOD = :ARTCOD
                      INTO :ARTDES
    END-EXEC.

    IF SQLSTATE NOT = "00000"
       MOVE " No existe ..." TO "Caption" OF INFORMACION3
    ELSE
       MOVE ARTDES TO "Caption" OF INFORMACION3
    END-IF.

    INVOKE CAMPO4 "SetFocus".
    EXIT PROGRAM.
```

Evento *Click* de botón BTCOD3:

```
ENVIRONMENT     DIVISION.
DATA            DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE       DIVISION.

    IF "Text" OF CAMPO3 = SPACES
       MOVE SPACES TO AHIVA15T
    ELSE
       MOVE "Text" OF CAMPO3 TO AHIVA15T
    END-IF.

    INVOKE POW-SELF "CallForm" USING "F-CON-ARTICULOS".

    IF AHIVA15T = SPACES
       INVOKE BTCOD3 "SetFocus"
       EXIT PROGRAM
    END-IF.
```

```
MOVE AHIVA15T TO "Text" OF CAMPO3.

IF POW-TEXT OF CAMPO1 > POW-TEXT OF CAMPO3
   MOVE " El primero no puede ser mayor que el segundo ..."
        TO "Caption" OF INFORMACION3
   MOVE POW-COLOR-DARKRED TO "ForeColor" OF INFORMACION3
   INVOKE CAMPO3 "SetFocus"
   EXIT PROGRAM
END-IF.

MOVE POW-COLOR-BLACK TO "ForeColor" OF INFORMACION3.

MOVE AHIVA15T TO ARTCOD.

EXEC SQL
     SELECT ARTDES FROM ARTICULOS
            WHERE ARTCOD = :ARTCOD
                  INTO :ARTDES
END-EXEC.

IF SQLSTATE NOT = "00000"
   MOVE " No Existe ..." TO "Caption" OF INFORMACION3
ELSE
   MOVE ARTDES TO "Caption" OF INFORMACION3
END-IF.

INVOKE CAMPO4 "SetFocus".
EXIT PROGRAM.
```

Evento *Return* de CAMPO4:

```
ENVIRONMENT     DIVISION.
DATA            DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE       DIVISION.

     IF POW-TEXT OF CAMPO4 = SPACES
        MOVE POW-COLOR-BLACK TO "ForeColor" OF INFORMACION4
        MOVE " ... Hasta la última" TO "Caption" OF INFORMACION4
        MOVE SPACES TO POW-TEXT OF CAMPO4
        MOVE HIGH-VALUES TO WCOD40

        IF "Enabled" OF CHECK1 = 1
           INVOKE CHECK1 "SetFocus"
           EXIT PROGRAM
        ELSE
           INVOKE BTLISTAR "SetFocus"
           EXIT PROGRAM
        END-IF

     END-IF.

     IF POW-TEXT OF CAMPO2 > POW-TEXT OF CAMPO4
        MOVE " El primero no puede ser mayor que el segundo ..."
             TO "Caption" OF INFORMACION4
        MOVE POW-COLOR-DARKRED TO "ForeColor" OF INFORMACION4
        INVOKE CAMPO4 "SetFocus"
        EXIT PROGRAM
     END-IF.

     MOVE POW-COLOR-BLACK TO "ForeColor" OF INFORMACION4.

     MOVE POW-TEXT OF CAMPO4 TO FAMCOD.

     READ FAMILIAS NO LOCK
          INVALID KEY
```

```
                MOVE " No Existe ..." TO "Caption" OF INFORMACION4
          NOT INVALID KEY
              MOVE FAMDES TO "Caption" OF INFORMACION4
     END-READ.

     IF "Enabled" OF CHECK1 = 1
        INVOKE CHECK1 "SetFocus"
        EXIT PROGRAM
     ELSE
        INVOKE BTLISTAR "SetFocus"
        EXIT PROGRAM
     END-IF.
```

Evento *Click* de botón BTCOD4:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.

     MOVE SPACES TO AHIVA3T.
     INVOKE POW-SELF "CallForm" USING "F-CON-FAMILIAS" "M-FAMILIAS".

     IF AHIVA3T = SPACES
        INVOKE BTCOD4 "SetFocus"
        EXIT PROGRAM
     END-IF.

     MOVE AHIVA3T TO "Text" OF CAMPO4.

     IF POW-TEXT OF CAMPO2 > POW-TEXT OF CAMPO4
        MOVE " El primero no puede ser mayor que el segundo ..."
             TO "Caption" OF INFORMACION4
        MOVE POW-COLOR-DARKRED TO "ForeColor" OF INFORMACION4
        INVOKE CAMPO4 "SetFocus"
        EXIT PROGRAM
     END-IF.

     MOVE POW-COLOR-BLACK TO "ForeColor" OF INFORMACION4.
     MOVE AHIVA3T TO FAMCOD.

     READ FAMILIAS NO LOCK
          INVALID KEY
                  MOVE " No Existe ..." TO "Caption" OF INFORMACION4
          NOT INVALID KEY
              MOVE FAMDES TO "Caption" OF INFORMACION4
     END-READ.

     INVOKE CAMPO3 "SetFocus".
     EXIT PROGRAM.
```

Evento *Click* de *CmCheck* CHECK5:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.
*
     IF POW-CHECK OF CHECK5 = POW-ON
        MOVE 1 TO "Enabled" OF CAMPO10
     ELSE
        MOVE 0 TO "Enabled" OF CAMPO10
     END-IF.
```

Evento *Click* de *CmCheck* CHECK6:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.
*
     IF POW-CHECK OF CHECK6 = POW-ON
        MOVE 1       TO "Enabled" OF CAMPO11
        MOVE 1       TO "Enabled" OF CAMPO12
        MOVE 1       TO "Enabled" OF CmStatic7
        MOVE 1       TO "Enabled" OF BTFIC
        MOVE 1       TO "Enabled" OF CHECK1
        MOVE 1       TO "Enabled" OF CHECK2
        MOVE 1       TO "Enabled" OF CHECK3
        MOVE 1       TO "Enabled" OF CHECK4
        MOVE 1       TO "Enabled" OF CHECK5
        MOVE POW-OFF TO POW-CHECK OF CHECK7
        MOVE 0       TO "Enabled" OF CHECK8
        MOVE 0       TO "Enabled" OF CHECK9
        MOVE 0       TO "Enabled" OF BTIMP
     ELSE
        MOVE 0 TO "Enabled" OF CAMPO11
        MOVE 0 TO "Enabled" OF CAMPO12
        MOVE 0 TO "Enabled" OF CmStatic7
        MOVE 0 TO "Enabled" OF BTFIC
        MOVE 1 TO "Enabled" OF CHECK8
        MOVE 1 TO "Enabled" OF CHECK9
        MOVE 1 TO "Enabled" OF BTIMP
     END-IF.
```

Evento *Return* de CAMPO11:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.
     IF "Text" OF CAMPO11 = SPACES
        MOVE "El Nombre del Fichero no puede estar Vacío ..."
             TO "StatusText" OF POW-SELF
        INVOKE CAMPO11 "SetFocus"
        EXIT PROGRAM
     ELSE
        MOVE SPACES TO "StatusText" OF POW-SELF
        INVOKE CAMPO12 "SetFocus"
        EXIT PROGRAM
     END-IF.
```

Evento *Click* de botón BTFIC:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.

     MOVE SPACES TO FILE-TEXT
     INVOKE POW-SELF "GetFileName"
            USING FILE-TEXT
                  "Por favor, Indique el nombre del Archivo"
```

```
                "*.txt    (Texto    Plano)|*.txt|*.csv    (Formato    de    valores
separados)|*.csv|*.tex (Texto Plano)|*.tex|*.* (Todos los Archivos)|*.*"
                POW-CDOPEN
            RETURNING RETORNO
     END-INVOKE.

     MOVE FILE-TEXT TO "Text" OF CAMPO11.
     INVOKE CAMPO11 "SetFocus".
     EXIT PROGRAM.
```

Evento *Return* de CAMPO12:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.
     IF "Text" OF CAMPO12 = SPACES
        INVOKE POW-SELF "Alarm"
        ADD POW-DMICONWARNING POW-DMYESNO POW-DMDEFBUTTON2 GIVING ESTILO
        INVOKE POW-SELF "DisplayMessage"
               USING "No se aconseja como separador el ESPACIO, ¿Desea cambiarlo?"
                     WS-ATENCION
                     ESTILO
               RETURNING RETORNO
        END-INVOKE
        IF RETORNO = POW-DMRYES
           INVOKE CAMPO12 "SetFocus"
           EXIT PROGRAM
        END-IF
     END-IF.

     INVOKE BTLISTAR "SetFocus".
     EXIT PROGRAM
```

Evento *Click* de *CmCheck* CHECK7:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.
     IF POW-CHECK OF CHECK7 = POW-ON
        MOVE 0       TO "Enabled" OF CHECK1
        MOVE 0       TO "Enabled" OF CHECK2
        MOVE 0       TO "Enabled" OF CHECK3
        MOVE 0       TO "Enabled" OF CHECK4
        MOVE 0       TO "Enabled" OF CHECK5
        MOVE POW-OFF TO POW-CHECK OF CHECK6
        MOVE 0       TO "Enabled" OF CAMPO11
        MOVE 0       TO "Enabled" OF CAMPO12
        MOVE 0       TO "Enabled" OF CmStatic7
        MOVE 0       TO "Enabled" OF BTFIC
        MOVE 0       TO "Enabled" OF CHECK8
        MOVE 0       TO "Enabled" OF CHECK9
        MOVE 0       TO "Enabled" OF BTIMP
     ELSE
        MOVE 1       TO "Enabled" OF CHECK1
        MOVE 1       TO "Enabled" OF CHECK2
        MOVE 1       TO "Enabled" OF CHECK3
        MOVE 1       TO "Enabled" OF CHECK4
        MOVE 1       TO "Enabled" OF CHECK5
        MOVE 1       TO "Enabled" OF CHECK8
        MOVE 1       TO "Enabled" OF CHECK9
        MOVE 1       TO "Enabled" OF BTIMP
     END-IF.
```

Evento *Click* de botón BTIMP:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.
     INVOKE POW-SELF "CallForm" USING "F-PRINTER" "M-PRINTER".

     IF WSPRINTER NOT = SPACES
        MOVE SPACES TO WTEXT
        MOVE SPACES TO "StatusText" OF POW-SELF
        STRING "Impresora seleccionada: "
               WSPRINTER
               DELIMITED BY SIZE INTO WTEXT
        END-STRING
        MOVE WTEXT TO "StatusText" OF POW-SELF
     ELSE
        MOVE SPACES TO "StatusText" OF POW-SELF
     END-IF.
```

Evento *Click* de botón BTLISTAR:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
*
 01  file-details.
     02 file-size             PIC 9(18) BINARY.
     02 file-date.
        03 d_ay                PIC 9(4)  BINARY.
        03 month               PIC 9(4)  BINARY.
        03 year                PIC 9(4)  BINARY.
     02 file-time.
        03 hour                PIC 9(4)  BINARY.
        03 minute              PIC 9(4)  BINARY.
        03 second              PIC 9(4)  BINARY.
        03 1-100-second        PIC 9(4)  BINARY.
  01 status-code               PIC S9(4) COMP-5.
*
  01  WSEP                     PIC X VALUE SPACES.
  01  TEXTO                    PIC X(300).
  01  TRES-EDI                 PIC ZZ9,99.
  01  SEIS-EDI                 PIC ---.--9.
  01  SIETE-EDI                PIC --.---.--9,999.
*
  01  HORA1.
      03  HH                   PIC 99.
      03                       PIC X VALUE ":".
      03  MM                   PIC 99.
      03                       PIC X VALUE "'".
      03  SS                   PIC 99.
  77  CONTA-EDI                PIC ZZZ9.
  77  P                        PIC 9999.   *> PUNTERO PARA EL STRING.
  77  B1                       PIC X(136) VALUE SPACES.
  77  B2                       PIC X(243) VALUE SPACES.
 PROCEDURE       DIVISION.

     IF POW-TEXT OF CAMPO1 = SPACES
        MOVE " Desde el primero ..." TO "Caption" OF INFORMACION1
        MOVE SPACES TO POW-TEXT OF CAMPO1
        MOVE SPACES TO WCOD10
```

```
      ELSE
         MOVE POW-TEXT OF CAMPO1 TO WCOD10
      END-IF.

      IF POW-TEXT OF CAMPO2 = SPACES
         MOVE " Desde la primera ..." TO "Caption" OF INFORMACION2
         MOVE SPACES TO POW-TEXT OF CAMPO2
         MOVE SPACES TO WCOD30
      ELSE
         MOVE POW-TEXT OF CAMPO2 TO WCOD30
      END-IF.

      IF POW-TEXT OF CAMPO3 = SPACES
         MOVE " ... Hasta el último" TO "Caption" OF INFORMACION3
         MOVE HIGH-VALUES TO WCOD20
      ELSE
         MOVE POW-TEXT OF CAMPO3 TO WCOD20
      END-IF.

      IF WCOD10 > WCOD20
         MOVE " El primero no puede ser mayor que el segundo ..."
              TO "Caption" OF INFORMACION3
         MOVE POW-COLOR-DARKRED TO "ForeColor" OF INFORMACION3
         INVOKE CAMPO3 "SetFocus"
         EXIT PROGRAM
      END-IF.

      IF POW-TEXT OF CAMPO4 = SPACES
         MOVE " ... Hasta la última" TO "Caption" OF INFORMACION4
         MOVE HIGH-VALUES TO WCOD40
      ELSE
         MOVE POW-TEXT OF CAMPO4 TO WCOD40
      END-IF.

      IF WCOD30 > WCOD40
         MOVE " El primero no puede ser mayor que el segundo ..."
              TO "Caption" OF INFORMACION4
         MOVE POW-COLOR-DARKRED TO "ForeColor" OF INFORMACION4
         INVOKE CAMPO4 "SetFocus"
         EXIT PROGRAM
      END-IF.

      IF POW-CHECK OF CHECK6 = POW-ON
         AND "Text" OF CAMPO11 = SPACES
             MOVE "El Nombre del Fichero no puede estar Vacío ..."
                  TO "StatusText" OF POW-SELF
             INVOKE CAMPO11 "SetFocus"
             EXIT PROGRAM
      ELSE
         MOVE SPACES TO "StatusText" OF POW-SELF
      END-IF.
*
 POSICIONO.

      MOVE 0 TO CONTASQL.

      EXEC SQL
           SELECT COUNT(ARTCOD) FROM ARTICULOS
                  WHERE BINARY ARTCOD BETWEEN :WCOD10 AND :WCOD20
                    AND BINARY ARTFAM BETWEEN :WCOD30 AND :WCOD40
                    AND ARTELI <> 'S'
                        INTO :CONTASQL
      END-EXEC.

      IF CONTASQL = 0
         ADD POW-DMICONEXCLAMATION POW-DMOK GIVING ESTILO
         INVOKE POW-SELF "DisplayMessage"
                USING "*ERROR: No hay registros con esos parámetros ..."
```

```
                    WS-ATENCION
                    ESTILO
        END-INVOKE
        INVOKE CAMPO1 "SetFocus"
        EXIT PROGRAM
     END-IF.
     MOVE CONTASQL TO CONTA.
*
 FIN-LEER.
     IF POW-CHECK OF CHECK7 = POW-ON      *> A Excel
        CALL "CARGAR"
        INVOKE CAMPO1 "SetFocus"
        EXIT PROGRAM
     END-IF.

     IF POW-CHECK OF CHECK6 = POW-OFF     *> Si no es a Excel y no es a Fichero, sólo
queda que sea por impresora
        CALL "POR-IMPRESORA"
        INVOKE CAMPO1 "SetFocus"
        EXIT PROGRAM
     ELSE                              *> De lo contario es A Fichero
        MOVE "Text" OF CAMPO11 TO FILE-TEXT
        MOVE "Text" OF CAMPO12 TO WSEP
     END-IF.

     EXEC SQL
          DECLARE cursor_n CURSOR FOR
                  SELECT * FROM ARTICULOS
                           WHERE BINARY ARTCOD BETWEEN :WCOD10 AND :WCOD20
                             AND BINARY ARTFAM BETWEEN :WCOD30 AND :WCOD40
                             AND ARTELI <> 'S'
     END-EXEC.

     MOVE 1     TO "Visible"      OF PROGRESO.
     MOVE CONTA TO "Max"          OF PROGRESO.
     MOVE 11    TO "MousePointer" OF POW-SELF.

     IF POW-CHECK OF CHECK1 = POW-ON
        MOVE FUNCTION CURRENT-DATE TO FECHA
        MOVE CORR FECHA TO FECHA1
     END-IF.

     IF POW-CHECK OF CHECK2 = POW-ON
        ACCEPT HORA FROM TIME
     END-IF.

     IF POW-CHECK OF CHECK4 = POW-ON
        PERFORM MONTAR-CONDICIONES
     ELSE
        MOVE SPACES TO WTEXT
     END-IF.

     CALL "CBL_CHECK_FILE_EXIST2"
          USING FILE-TEXT
                file-details
          RETURNING status-code
     END-CALL.

     IF status-code = 0
        INVOKE POW-SELF "Alarm"
        MOVE SPACES TO TEXTO
        COMPUTE P = FUNCTION STORED-CHAR-LENGTH (FILE-TEXT)
        STRING "El fichero "
               FILE-TEXT (1:P)
               SALTO
               "ya existe, ¿Reemplazar?"
               DELIMITED BY SIZE INTO TEXTO
        END-STRING
```

```
   ADD POW-DMICONQUESTION POW-DMYESNO POW-DMDEFBUTTON2 GIVING ESTILO

   INVOKE POW-SELF "DisplayMessage"
          USING TEXTO
                WS-ATENCION
                ESTILO
          RETURNING RETORNO
   END-INVOKE

   IF RETORNO = POW-DMRNO
      MOVE 0 TO "Visible"      OF PROGRESO
      MOVE 0 TO "MousePointer" OF POW-SELF
      INVOKE CAMPO11 "SetFocus"
      EXIT PROGRAM
   END-IF
END-IF.

OPEN OUTPUT FICHERO.

IF STFIC = "35"
   INVOKE POW-SELF "Alarm"
   ADD POW-DMICONERROR POW-DMOK GIVING ESTILO
   INVOKE POW-SELF "DisplayMessage"
          USING "Error [35] al crear el fichero de texto ..."
                WS-ATENCION
                ESTILO
   END-INVOKE
   INVOKE CAMPO1 "SetFocus"
   EXIT PROGRAM
END-IF.

 *>
 *> CABECERA del LISTADO (Sólo 1 vez)
 *>

MOVE SPACES TO FICTEXT.
MOVE "        ** FICHERO de ARTÍCULOS **" TO FICTEXT.
WRITE FICTEXT.
MOVE SPACES TO FICTEXT.
MOVE 3 TO P.

IF POW-CHECK OF CHECK1 = POW-ON      *> FECHA
   STRING "Fecha: "
          FECHA1
          "   "  DELIMITED BY SIZE INTO FICTEXT
                 WITH POINTER P
   END-STRING
END-IF.

IF POW-CHECK OF CHECK2 = POW-ON      *> HORA
   MOVE CORR HORA TO HORA1
   STRING "Hora: "
          HORA1
          "   "  DELIMITED BY SIZE INTO FICTEXT
                 WITH POINTER P
   END-STRING
END-IF.

IF POW-CHECK OF CHECK3 = POW-ON      *> PC
   MOVE 0 TO P
   COMPUTE P = 80 - (FUNCTION STORED-CHAR-LENGTH (WMAQ))
               - 5
   END-COMPUTE
   STRING "PC: "
          WMAQ   DELIMITED BY SIZE INTO FICTEXT
                 WITH POINTER P
```

```
           END-STRING
        END-IF.

        WRITE FICTEXT.

        IF POW-CHECK OF CHECK4 = POW-ON
           MOVE SPACES TO FICTEXT            *> CONDICIONES DEL LISTADO
           WRITE FICTEXT FROM WTEXT
        END-IF.

        IF POW-CHECK OF CHECK5 = POW-ON      *> REFERENCIA
           MOVE SPACES TO WTEXT FICTEXT
           MOVE "Text" OF CAMPO10 TO WTEXT
           STRING "Referencia: "
                  WTEXT DELIMITED BY SIZE INTO FICTEXT
           END-STRING
           WRITE FICTEXT
        END-IF.

        MOVE SPACES TO FICTEXT.
        WRITE FICTEXT.

        STRING     "CÓDIGO                                                    DESCRIPCIÓN
FAM Descripción "
                   "Familia                 Provee.  I.V.A.          Precio 1          Precio 2
Precio 3 O B S E R V A C I O N E S"
                   B1
                   "A Ruta IMAGEN 1"
                   B2
                   "Ruta IMAGEN 2"
                   B2
                   "Ruta IMAGEN 3"
                   B2
                   "Ruta IMAGEN 4"
                   B2
                   "Ruta IMAGEN 5"
                   B2
                   "Ruta IMAGEN 6"
                   B2
                   "Fecha ALTA Fec.U.Modi"
                   DELIMITED BY SIZE INTO FICTEXT
        END-STRING.

        WRITE FICTEXT.
        MOVE ALL "=" TO FICTEXT.
        WRITE FICTEXT.

         *>
         *> FIN CABECERA del LISTADO
         *>
*
 LEO.
        EXEC SQL
             OPEN cursor_n
        END-EXEC.

        PERFORM CARGAR THRU FIN-CARGAR UNTIL SQLSTATE = "02000".

        EXEC SQL
             CLOSE cursor_n
        END-EXEC.
*
 FIN-LEO.
        CLOSE FICHERO.
        ADD POW-DMICONINFORMATION POW-DMYESNO GIVING ESTILO.
        MOVE SPACES TO WTEXT.
        MOVE CONTA TO CONTA-EDI.
```

```
      STRING "Proceso finalizado, se han pasado "
             CONTA-EDI
             " Registros"
             SALTO
             "¿Desea Abrir el Fichero ahora?"
             SALTO
             " "
             DELIMITED BY SIZE INTO WTEXT
      END-STRING.

      INVOKE POW-SELF "DisplayMessage"
             USING WTEXT
                   WS-ATENCION
                   ESTILO
             RETURNING RETORNO
      END-INVOKE.

      IF RETORNO = POW-DMRYES
         MOVE SPACES TO FICTEXT
         STRING "notepad.exe "
                FILE-TEXT
                DELIMITED BY SIZE INTO FICTEXT
         END-STRING
         INVOKE POW-SELF "Execute" USING FICTEXT POW-SWSHOWMAXIMIZED
      END-IF.

      MOVE 0 TO "Visible"      OF PROGRESO.
      MOVE 0 TO "MousePointer" OF POW-SELF.
      INVOKE CAMPO1 "SetFocus".
      EXIT PROGRAM.
*
 CARGAR.

      EXEC SQL
           FETCH cursor_n
                 INTO :ARTCOD,
                      :ARTDES,
                      :ARTFAM,
                      :ARTPRO,
                      :ARTIVA,
                      :ARTPRE1,
                      :ARTPRE2,
                      :ARTPRE3,
                      :ARTOBS,
                      :ARTACT,
                      :ARTIMG1,
                      :ARTIMG2,
                      :ARTIMG3,
                      :ARTIMG4,
                      :ARTIMG5,
                      :ARTIMG6,
                      :ARTFEC,
                      :ARTFEC1,
                      :ARTELI
      END-EXEC.

      IF SQLSTATE = "02000"
         GO FIN-CARGAR
      END-IF.

      MOVE SPACES TO FICTEXT.

      MOVE ARTFAM TO FAMCOD.
      READ FAMILIAS NO LOCK
           INVALID KEY
                   MOVE " No existe en fichero ..." TO FAMDES.

      MOVE ARTIVA  TO TRES-EDI.
```

```
MOVE ARTPRO  TO SEIS-EDI.
MOVE ARTPRE1 TO SIETE-EDI.

STRING ARTCOD
       WSEP
       ARTDES
       WSEP
       ARTFAM
       WSEP
       FAMDES
       WSEP
       SEIS-EDI
       WSEP
       TRES-EDI
       WSEP
       SIETE-EDI
       DELIMITED BY SIZE INTO FICTEXT
END-STRING.

COMPUTE P = FUNCTION STORED-CHAR-LENGTH (FICTEXT) + 1.

MOVE ARTPRE2 TO SIETE-EDI.

STRING WSEP
       SIETE-EDI
       DELIMITED BY SIZE INTO FICTEXT WITH POINTER P
END-STRING.

COMPUTE P = FUNCTION STORED-CHAR-LENGTH (FICTEXT) + 1.

MOVE ARTPRE3 TO SIETE-EDI.

STRING WSEP
       SIETE-EDI
       WSEP
       ARTOBS
       WSEP
       ARTACT
       WSEP
       ARTIMG1
       WSEP
       ARTIMG2
       WSEP
       ARTIMG3
       WSEP
       ARTIMG4
       WSEP
       ARTIMG5
       WSEP
       ARTIMG6
       WSEP
       DELIMITED BY SIZE INTO FICTEXT WITH POINTER P
END-STRING.

COMPUTE P = FUNCTION STORED-CHAR-LENGTH (FICTEXT) + 1.

MOVE ARTFEC TO FECHA.
MOVE CORR FECHA TO FECHA-EDI.

STRING FECHA-EDI
       DELIMITED BY SIZE INTO FICTEXT WITH POINTER P
END-STRING.

COMPUTE P = FUNCTION STORED-CHAR-LENGTH (FICTEXT) + 1.

MOVE ARTFEC1 TO FECHA.
MOVE CORR FECHA TO FECHA-EDI.

STRING WSEP
```

```
            FECHA-EDI
            DELIMITED BY SIZE INTO FICTEXT WITH POINTER P
     END-STRING.
     WRITE FICTEXT.

     INVOKE PROGRESO "ProgressStep".
*
 FIN-CARGAR.
     EXIT.
*
 MONTAR-CONDICIONES.
     MOVE SPACES TO WTEXT.

     STRING "Condiciones: Desde "
            WCOD10
            "  Hasta "
            WCOD20
            "." DELIMITED BY SIZE INTO WTEXT
     END-STRING.
```

Evento *Click* de botón BTSALIR:

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.
     INVOKE POW-SELF "Deactivate".
```

Subrutinas del programa “Listados”

Subrutina “A-EXCEL”

```
*
*
* ----- Mil Gracias por el código RAIPINTO
*
*
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  EXCEL              OBJECT REFERENCE OLE.
 01  WORKBOOK           OBJECT REFERENCE OLE.
 01  SHEETS             OBJECT REFERENCE OLE.
 01  WORKSHEET          OBJECT REFERENCE OLE.
 01  CELL               OBJECT REFERENCE OLE.
 01  COLUMNS            OBJECT REFERENCE OLE.
 01  WCOLUMN            OBJECT REFERENCE OLE.
 01  OBJRANGE           OBJECT REFERENCE OLE.
 01  FITRANGE           OBJECT REFERENCE OLE.
*
 01  ARRAYOBJ OBJECT REFERENCE COM-ARRAY.
 01  LONG-INT-TYPE                 PIC S9(9) COMP-5 VALUE 12.
 01  ARRAY-DIMENSION               PIC S9(9) COMP-5 VALUE 2.
 01  AXIS-1                        PIC S9(9) COMP-5 VALUE 22.
 01  AXIS-2                        PIC S9(9) COMP-5 VALUE 22.
*
 01  APPLICATION                   PIC X(20) VALUE "EXCEL.APPLICATION".
 01  OLE-TRUE                      PIC 1(1)  BIT VALUE B"1".
 01  FILLER                        PIC 1(7)  BIT.
 01  OLE-FALSE                     PIC 1(1)  BIT VALUE B"0".
 01  FILLER                        PIC 1(7)  BIT.
 01  ARRAY-ROW                     PIC S9(9) COMP-5.
 01  ARRAY-COL                     PIC S9(9) COMP-5.
 01  VAL                           PIC X(256).
 01  S-INDEX                       PIC S9(4) COMP-5 VALUE 1.
 01  LINHA                         PIC X(1024).
 01  OLE-ERR-METHOD                PIC X(256).
 01  OLE-ERR-INFO.
     03  OLE-ERR-TYPE              PIC X(001).
     03  OLE-ERR-WCODE             PIC X(002).
     03  ROLE-ERR-WCODE REDEFINES OLE-ERR-WCODE  PIC S9(04) COMP-5.
     03  OLE-ERR-SCODE             PIC X(004).
     03  ROLE-ERR-SCODE REDEFINES OLE-ERR-SCODE PIC S9(09) COMP-5.
*
 01  XARRCOL                  PIC X(026)  VALUE "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
 01  ARRCOLS  REDEFINES XARRCOL.
     03   ARRCOL OCCURS 26    PIC X(001).
*
 01  FORMATNUM                PIC X(9)  VALUE "#.##0,00".
 01  FORMATDATE               PIC X(12) VALUE "aaaa-mm-dd".
 01  NRCOLS                   PIC S9(9) COMP-5   VALUE 1.
 01  INIROW                   PIC S9(9) COMP-5   VALUE 2.
 01  INICOL                   PIC S9(9) COMP-5   VALUE 1.
 01  ENDROW                   PIC S9(9) COMP-5   VALUE 2.
 01  ENDCOL                   PIC S9(9) COMP-5   VALUE 22.
 01  INIRANGE                 PIC X(19)          VALUE " A0000002:AZ0000002".
 01  ROWVAL                   PIC 9(007).
 01  COLVAL                   PIC 9(007).
 01  PICSTRING                PIC X(020).
 01  PICDATE                  PIC X(10).
*
 01  IDX1                     PIC S9(5) COMP-5.
 01  IDX2                     PIC S9(5) COMP-5.
 01  IDX3                     PIC S9(5) COMP-5.
```

```
 01  IDX4                    PIC S9(5) COMP-5.
 01  IDX5                    PIC S9(5) COMP-5.
 01  WSLONG                  PIC S9(5) COMP-5.
*
 PROCEDURE DIVISION.
 DECLARATIVES.
 OLE-ERRO SECTION.
     USE AFTER EXCEPTION OLE-EX.
*
 END DECLARATIVES.
*
 MAIN SECTION.
 MAIN-00.
           IF "COUNT" OF View2 = ZERO
              GO TO MAIN-99
           END-IF.

           MOVE 11 TO "MousePointer" OF POW-SELF.
           MOVE 1  TO WSLONG.
           MOVE 1  TO S-INDEX.

           INVOKE OLE "CREATE-OBJECT" USING APPLICATION
                  RETURNING EXCEL
           END-INVOKE.

           INVOKE EXCEL "GET-WORKBOOKS"
                  RETURNING WORKBOOK
           END-INVOKE.

           INVOKE WORKBOOK "ADD"
                  RETURNING WORKBOOK
           END-INVOKE.

           INVOKE WORKBOOK "GET-WORKSHEETS"
                  RETURNING SHEETS
           END-INVOKE.

           INVOKE SHEETS "GET-ITEM"
                  USING S-INDEX
                  RETURNING WORKSHEET
           END-INVOKE.
*
           MOVE "ColumnCount" OF View2    TO IDX1.
*
           MOVE 1 TO IDX5.

           PERFORM VARYING IDX2 FROM 1 BY 1 UNTIL IDX2 > IDX1
                   IF "Width" OF "Columns"(IDX2) OF View2 > 5
                       MOVE "HeaderText" OF "Columns"(IDX2) OF View2 TO VAL
                       MOVE  WSLONG TO ARRAY-ROW
                       MOVE IDX5 TO ARRAY-COL
                       INVOKE WORKSHEET "GET-CELLS"
                              USING ARRAY-ROW ARRAY-COL
                              RETURNING CELL
                       END-INVOKE
                       INVOKE CELL "SET-VALUE"
                              USING VAL
                       END-INVOKE
                       ADD 1                              TO IDX5
                   END-IF
           END-PERFORM.
*
           ADD 1                          TO WSLONG.
           MOVE "Count" OF View2          TO IDX1.
           MOVE "ColumnCount" OF View2    TO IDX2.
*
           MOVE IDX1 TO AXIS-1.
```

```
           MOVE IDX2 TO AXIS-2.

           INVOKE COM-ARRAY "NEW"
                  USING LONG-INT-TYPE
                        ARRAY-DIMENSION
                        AXIS-1
                        AXIS-2
                  RETURNING ARRAYOBJ
           END-INVOKE.

           INVOKE POW-SELF "THRUEVENTS".
*
*  VAI CARREGAR O ARRAY BIDINENSIONAL OLE-ARRAY A PARTIR DA LISTVIEW
*
           PERFORM VARYING IDX3 FROM 1 BY 1 UNTIL IDX3 > IDX1
                   MOVE 1 TO IDX5
                   PERFORM VARYING IDX4 FROM 1 BY 1 UNTIL IDX4 > IDX2
                           IF "Width" OF "Columns"(IDX4) OF View2 > 5
                              MOVE "Text"(IDX4) OF "ListItems"(IDX3) OF View2   TO
LINHA
                              INVOKE ARRAYOBJ "SET-DATA"
                                     USING LINHA IDX3 IDX5
                              END-INVOKE
                              ADD 1 TO IDX5
                           END-IF
                   END-PERFORM
                   IF IDX3 = 5000 OR = 10000 OR = 15000 OR = 20000 OR = 250000 OR =
30000
                      OR = 35000 OR = 40000
                         INVOKE POW-SELF "THRUEVENTS"
                   END-IF
           END-PERFORM.
*
************  CALCULAR O "RANGE" DA FOLHA TODA PARA ENVIAR O ARRAY DUMA SO VEZ
*  A PRIMEIRA CELULA É SEMPRE "A00002"
           MOVE 1 TO ROWVAL.
           ADD  1 TO ROWVAL .
           MOVE ROWVAL TO INIRANGE(3:7).

           EVALUATE IDX2
                    WHEN 183 THRU 208
                         SUBTRACT 182 FROM IDX2 GIVING COLVAL
                         MOVE "G"             TO INIRANGE(11:1)
                         MOVE ARRCOL(COLVAL) TO INIRANGE(12:1)
                    WHEN 157 THRU 182
                         SUBTRACT 156 FROM IDX2 GIVING COLVAL
                         MOVE "F"             TO INIRANGE(11:1)
                         MOVE ARRCOL(COLVAL) TO INIRANGE(12:1)
                    WHEN 131 THRU 156
                         SUBTRACT 130 FROM IDX2 GIVING COLVAL
                         MOVE "E"             TO INIRANGE(11:1)
                         MOVE ARRCOL(COLVAL) TO INIRANGE(12:1)
                    WHEN 105 THRU 130
                         SUBTRACT 104 FROM IDX2 GIVING COLVAL
                         MOVE "D"             TO INIRANGE(11:1)
                         MOVE ARRCOL(COLVAL) TO INIRANGE(12:1)
                    WHEN 79 THRU 104
                         SUBTRACT 78 FROM IDX2 GIVING COLVAL
                         MOVE "C"             TO INIRANGE(11:1)
                         MOVE ARRCOL(COLVAL) TO INIRANGE(12:1)
                    WHEN 53 THRU 78
                         SUBTRACT 52 FROM IDX2 GIVING COLVAL
                         MOVE "B"             TO INIRANGE(11:1)
                         MOVE ARRCOL(COLVAL) TO INIRANGE(12:1)
                    WHEN 27 THRU 52
                         SUBTRACT 26 FROM IDX2 GIVING COLVAL
                         MOVE "A"             TO INIRANGE(11:1)
                         MOVE ARRCOL(COLVAL) TO INIRANGE(12:1)
```

```
                    WHEN 1 THRU 26
                         MOVE " "          TO INIRANGE(11:1)
                         MOVE ARRCOL(IDX2) TO INIRANGE(12:1)
           END-EVALUATE

           MOVE IDX1 TO ROWVAL.
           ADD 1 TO ROWVAL.
           MOVE ROWVAL TO INIRANGE(13:7).
           INVOKE WORKSHEET "GET-RANGE"
                  USING INIRANGE
                  RETURNING OBJRANGE
           END-INVOKE.
           INVOKE OBJRANGE "SET-VALUE"
                  USING ARRAYOBJ
           END-INVOKE.
*
 MAIN-80.
*
* UTILIZANDO O RANGE QUE FOI DAS COLUNAS/LINHAS UTILIZADAS VAI FAZER O AUTOFIT
(ALARGAR AS COLUNAS)
*
            INVOKE WORKSHEET "GET-USEDRANGE"
                   RETURNING OBJRANGE
            END-INVOKE.
            INVOKE OBJRANGE "GET-ENTIRECOLUMN"
                   RETURNING FITRANGE
            END-INVOKE.
            INVOKE FITRANGE "AUTOFIT".
            MOVE 0 TO "MousePointer" OF POW-SELF.
*  DESTRUIR OS OBJECTOS UTILIZADOS PARA LIBERTAR MEMORIA
            INVOKE EXCEL "SET-VISIBLE" USING OLE-TRUE.
*           INVOKE EXCEL "QUIT".
            SET CELL      TO NULL.
            SET COLUMNS   TO NULL.
            SET FITRANGE  TO NULL.
            SET OBJRANGE  TO NULL.
            SET WORKSHEET TO NULL.
            SET SHEETS    TO NULL.
            SET WORKBOOK  TO NULL.
            SET EXCEL     TO NULL.
 MAIN-99.
            EXIT PROGRAM.
```

Subrutina “CARGAR”

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01   POW-PCMLIST              OBJECT REFERENCE POW-CLISTITEM.
 01   WK-IDX                   PIC S9(9) COMP-5.
*
 PROCEDURE       DIVISION.
*
 POSICIONO.
         INVOKE VIEW2 "Clear".
*
     MOVE "Text" OF CAMPO1 TO WCOD10.
     MOVE "Text" OF CAMPO2 TO WCOD30.
     MOVE "Text" OF CAMPO3 TO WCOD20.
     MOVE "Text" OF CAMPO4 TO WCOD40.
     MOVE 0                TO WK-IDX.

     EXEC SQL
          SELECT COUNT(ARTCOD) FROM ARTICULOS
                 WHERE BINARY ARTCOD BETWEEN :WCOD10 AND :WCOD20
```

```
                    AND BINARY ARTFAM BETWEEN :WCOD30 AND :WCOD40
                    AND ARTELI <> 'S'
                        INTO :CONTASQL
     END-EXEC.

     IF CONTASQL = 0
        ADD POW-DMICONEXCLAMATION POW-DMOK GIVING ESTILO
        INVOKE POW-SELF "DisplayMessage"
               USING "*ERROR: No hay registros con esos parámetros ..."
                     WS-ATENCION
                     ESTILO
        END-INVOKE
        INVOKE CAMPO1 "SetFocus"
        EXIT PROGRAM
     END-IF.

     MOVE CONTASQL TO CONTA.
     MOVE 1     TO "Visible"      OF PROGRESO.
     MOVE CONTA TO "Max"          OF PROGRESO.
     MOVE 11    TO "MousePointer" OF POW-SELF.
*
 LEO.
     EXEC SQL
          DECLARE cursor_2 CURSOR FOR
                  SELECT * FROM ARTICULOS
                         WHERE BINARY ARTCOD BETWEEN :WCOD10 AND :WCOD20
                           AND BINARY ARTFAM BETWEEN :WCOD30 AND :WCOD40
                           AND ARTELI <> 'S'
     END-EXEC.

     EXEC SQL
          OPEN cursor_2
     END-EXEC.

     PERFORM CARGAR THRU FIN-CARGAR UNTIL SQLSTATE = "02000".

     EXEC SQL
          CLOSE cursor_2
     END-EXEC.
*
 FIN-LEO.
     CALL "A-EXCEL".

     MOVE 0 TO "Visible"      OF PROGRESO.
     MOVE 0 TO "MousePointer" OF POW-SELF.

     INVOKE BTSALIR "SetFocus".
*
     EXIT PROGRAM.
*
 CARGAR.

     INVOKE PROGRESO "ProgressStep".

     MOVE ARTFAM TO FAMCOD.
     READ FAMILIAS NO LOCK
          INVALID KEY
                  MOVE " No existe en fichero ..." TO FAMDES
     END-READ.

     INVOKE VIEW2 "Add" USING 1 1 RETURNING WK-IDX

     IF WK-IDX > 0 THEN
        MOVE "ListItems"(WK-IDX) OF VIEW2 TO POW-PCMLIST
     END-IF.

     MOVE ARTCOD          TO "Text"    (01) OF POW-PCMLIST.
     MOVE ARTDES        TO "Text"    (02) OF POW-PCMLIST.
     MOVE ARTFAM        TO "Text"    (03) OF POW-PCMLIST.
```

```
     MOVE FAMDES        TO "Text"    (04) OF POW-PCMLIST.
     MOVE ARTPRO        TO "Numeric" (05) OF POW-PCMLIST.
     MOVE ARTIVA        TO "Numeric" (06) OF POW-PCMLIST.
     MOVE ARTPRE1       TO "Numeric" (07) OF POW-PCMLIST.
     MOVE ARTPRE2       TO "Numeric" (08) OF POW-PCMLIST.
     MOVE ARTPRE3       TO "Numeric" (09) OF POW-PCMLIST.
     MOVE ARTOBS        TO "Text"    (10) OF POW-PCMLIST.
     MOVE ARTACT        TO "Text"    (11) OF POW-PCMLIST.
     MOVE ARTIMG1       TO "Text"    (12) OF POW-PCMLIST.
     MOVE ARTIMG2       TO "Text"    (13) OF POW-PCMLIST.
     MOVE ARTIMG3       TO "Text"    (14) OF POW-PCMLIST.
     MOVE ARTIMG4       TO "Text"    (15) OF POW-PCMLIST.
     MOVE ARTIMG5       TO "Text"    (16) OF POW-PCMLIST.
     MOVE ARTIMG6       TO "Text"    (17) OF POW-PCMLIST.
     MOVE ARTFEC        TO "Text"    (18) OF POW-PCMLIST.
     MOVE ARTFEC1       TO "Text"    (19) OF POW-PCMLIST.
*
 FIN-CARGAR.
     EXIT.
```

Subrutina "POR-IMPRESORA"

```
 ENVIRONMENT     DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT  PRINT-FILE          ASSIGN TO XPRINTER
             ORGANIZATION        IS SEQUENTIAL
             ACCESS MODE         IS SEQUENTIAL
             FORMAT              IS PRT-FORMAT
             GROUP               IS PRT-GROUP
             FILE STATUS         IS PRT-STATUS
                                    PRT-STATUS2.

     SELECT  RPT-IMPRESORA       ASSIGN TO NOM-ARCH
             ORGANIZATION        IS LINE SEQUENTIAL
             FILE STATUS         IS ST-XX.
*
 DATA            DIVISION.
 FILE SECTION.
 FD   PRINT-FILE.
      COPY  FLisArt1 OF XMDLIB.

 FD  RPT-IMPRESORA.
 01  IMPRESO.
     03 LINEA                       PIC X(254).
*
 WORKING-STORAGE SECTION.
 01  PRT-FORMAT                     PIC X(8).
 01  PRT-GROUP                      PIC X(8).
*
 01  C                              PIC 99     COMP-3.
 01  D                              PIC 99     COMP-3.
 01  LIN                            PIC 99     COMP-3.
 01  PAG                            PIC 999.
 01  bCANCELO                       PIC S9(4)  COMP-5 VALUE POW-FALSE.
 01  nTOTLIN                        PIC 9(2)   COMP-3 VALUE 58.
 01  NOM-ARCH                       PIC X(255) VALUE "C:\Articulos\Datos\Temp\A.RPT".
*> File Status de la impresora
 01  PRT-STATUS                     PIC XX.
 01  PRT-STATUS2.
           02 prt-sts1                  PIC XX.
           02 prt-sts2                  PIC XX.
                88 PRT-TODO-OK          VALUE "00".
                88 PRT-NOT-OPEN         VALUE "03".
```

```
                 88 PRT-NOMBRE-MAL       VALUE "24".
                 88 PRT-MAX-LINE         VALUE "62".
                 88 PRT-BTNCAN-PRTDLG    VALUE "9L".
                 88 PRT-PROCE-CAN        VALUE "92".
 01  ST-XX                          PIC X(02).
     88 TODO-OK         VALUE "00".
     88 CLAVE-DUP       VALUE "02".
     88 OKOPTIONAL      VALUE "05".
     88 FIN-ARCH        VALUE "10".
     88 REG-YA-ESTA     VALUE "22".
     88 REG-NO-ESTA     VALUE "23".
     88 ARCH-LIMIT      VALUE "24".
     88 NO-IMPRE        VALUE "30".
     88 ERROR-DOS       VALUE "30".
     88 ESTRUCTURA-MAL VALUE "39".
     88 ARCH-NO-ESTA    VALUE "35".
     88 NO-START        VALUE "46".
     88 ARCH-ABIERTO    VALUE "41".
     88 ARCH-LOCK       VALUE "93".
     88 LARGO-MAL       VALUE "04" "94" "97".
     88 CLAVE-MAL       VALUE "98".
     88 REG-LOCK        VALUE "99".
*
* PRT-NOMBRE-MAL "24": El nombre del campo o el grupo no existen en el
archivo(PowerFORM).
* PRT-BTN-CAN    "9L": El boton del Cancelar fue presionado en el cuadro de dialogo
*
*
 PROCEDURE       DIVISION.
 DECLARATIVES.
*
 ERROR-INPUT  SECTION.
         USE AFTER STANDARD ERROR PROCEDURE ON INPUT.
 ERROR-I-O    SECTION.
        USE AFTER STANDARD ERROR PROCEDURE ON I-O.
 ERROR-OUTPUT SECTION.
        USE AFTER STANDARD ERROR PROCEDURE ON OUTPUT.
 ERROR-EXTEND SECTION.
         USE AFTER STANDARD ERROR PROCEDURE ON EXTEND.
*
 END DECLARATIVES.
*
 UNICA SECTION.

 COMIENZO.
     MOVE 1          TO PAG LIN.
     MOVE POW-FALSE TO bCANCELO.
*
 ABRIR-IMPRESORA-RPT.
     OPEN OUTPUT RPT-IMPRESORA.

     IF POW-CHECK OF CHECK8 = POW-ON      *> Preview
        MOVE 'PREVIEW 2' TO LINEA
     ELSE
        MOVE 'PREVIEW 1' TO LINEA
     END-IF.

     WRITE IMPRESO.
*
*           Valores para PREVIEW:
*
*                 0: Depend on environment variable MEFTPRE.
*                 1: No preview.
*                 2: Asynchronous mode. Printing process will not wait for closing of
preview window.
*                 3: Synchronous mode. Printing process will wait for closing of
preview window.
*                      Default: 0
*
```

```
      IF POW-CHECK OF CHECK9 = POW-ON      *> Diálogo de Impresora
         MOVE 'SETPRTDIALOG 3' TO LINEA
      ELSE
         MOVE 'SETPRTDIALOG 1' TO LINEA
      END-IF.
*
*           Valores para SETPRTDIALOG:
*
*                 0: Depend on environment variable "MEFTDLG".
*                 1: No dialog box.
*                 2: Show dialog box.
*                 3: Show dialog box with value of printer information file.
*                       Default: 0
*
      WRITE IMPRESO.

      IF WSPRINTER NOT = SPACES      *> Han seleccionado una impresora
         COMPUTE C = FUNCTION STORED-CHAR-LENGTH (WSPRINTER)
         MOVE 'PRTNAME "' TO LINEA
         MOVE WSPRINTER TO LINEA (10:C)
         COMPUTE C = 1 + FUNCTION STORED-CHAR-LENGTH (LINEA)
         MOVE '"' TO LINEA (C:1)
         WRITE IMPRESO
      END-IF.

      MOVE 'PRTID "LISTADO de ARTÍCULOS"' TO LINEA.
      WRITE IMPRESO.
      MOVE 'PRTDIALG "Y"' TO LINEA.
      WRITE IMPRESO.
      CLOSE RPT-IMPRESORA.
*
 POSICIONO.

      EXEC SQL
           DECLARE cursor_3 CURSOR FOR
                   SELECT * FROM ARTICULOS
                          WHERE BINARY ARTCOD BETWEEN :WCOD10 AND :WCOD20
                            AND BINARY ARTFAM BETWEEN :WCOD30 AND :WCOD40
                            AND ARTELI <> 'S'
      END-EXEC.

      EXEC SQL
           OPEN cursor_3
      END-EXEC.
*
 ABRO-IMPRESORA.
            OPEN OUTPUT PRINT-FILE.

            IF PRT-BTNCAN-PRTDLG OR PRT-PROCE-CAN
               EXIT PROGRAM
            END-IF.

            IF NOT PRT-TODO-OK
                   ADD POW-DMOK POW-DMICONERROR GIVING ESTILO
                   STRING "Impresora devolvió error [ "
                        PRT-STS2 " ] OPEN IMPRESORA"
                        DELIMITED BY SIZE INTO WTEXT
                   INVOKE POW-SELF "DisplayMessage"
                        USING WTEXT
                              "Error"
                                ESTILO
                   END-INVOKE
                   EXIT PROGRAM
            END-IF.
*
       *>
       *> Comprobamos checks
       *>
*
```

```
      IF POW-CHECK OF CHECK1 = POW-ON            *> Fecha
         MOVE FUNCTION CURRENT-DATE TO FECHA
         MOVE CORR FECHA TO FECHA-EDI
         MOVE FECHA-EDI  TO P-FEC
         MOVE " "        TO EDIT-MODE OF ESTAT-001
         MOVE " "        TO EDIT-MODE OF P-FEC
         MOVE "Fecha:"   TO ESTAT-001
      ELSE
         MOVE "X" TO EDIT-MODE OF ESTAT-001
         MOVE "X" TO EDIT-MODE OF P-FEC
      END-IF.

      IF POW-CHECK OF CHECK2 = POW-ON            *> Hora
         ACCEPT HORA FROM TIME
         MOVE CORR HORA TO HORA-EDI
         MOVE HORA-EDI  TO PHOR
         MOVE " "       TO EDIT-MODE OF ESTAT-002
         MOVE " "       TO EDIT-MODE OF PHOR
         MOVE "Hora:"   TO ESTAT-002
      ELSE
         MOVE "X" TO EDIT-MODE OF ESTAT-002
         MOVE "X" TO EDIT-MODE OF PHOR
      END-IF.

      IF POW-CHECK OF CHECK3 = POW-ON            *> Pc
         COMPUTE C = FUNCTION STORED-CHAR-LENGTH (WMAQ)
         MOVE SPACES TO WTEXT
         MOVE WMAQ (1:C) TO WTEXT ((D + 2):C)
         COMPUTE C = FUNCTION STORED-CHAR-LENGTH (WTEXT)
         MOVE WTEXT TO ESTAT-003
         MOVE " " TO EDIT-MODE OF ESTAT-003
      ELSE
         MOVE "X" TO EDIT-MODE OF ESTAT-003
      END-IF.

      IF POW-CHECK OF CHECK4 = POW-ON            *> Condiciones
         PERFORM MONTAR-CONDICIONES
         MOVE " " TO EDIT-MODE OF PCON
         MOVE WTEXT TO PCON
      ELSE
         MOVE "X" TO EDIT-MODE OF PCON
      END-IF.

      IF POW-CHECK OF CHECK5 = POW-ON            *> Referencia
         MOVE " " TO EDIT-MODE OF PREF
         MOVE POW-TEXT OF CAMPO10 TO PREF
      ELSE
         MOVE "X" TO EDIT-MODE OF PREF
      END-IF.

      PERFORM CABECERA THRU FIN-CABECERA.
*
* ----- LEER FICHERO
*
 LEER.

      PERFORM CARGAR THRU FIN-CARGAR UNTIL SQLSTATE = "02000".

      EXEC SQL
           CLOSE cursor_3
      END-EXEC.
*
 FIN-LEER.
      MOVE 0 TO "MousePointer" OF POW-SELF.
      MOVE POW-FALSE TO bCANCELO.
      CLOSE PRINT-FILE.
      INVOKE BTLISTAR "SetFocus".
      EXIT PROGRAM.
*
```

```
CARGAR.

    EXEC SQL
         FETCH cursor_3
               INTO
                  :ARTCOD,
                      :ARTDES,
                      :ARTFAM,
                      :ARTPRO,
                      :ARTIVA,
                      :ARTPRE1,
                      :ARTPRE2,
                      :ARTPRE3,
                      :ARTOBS,
                      :ARTACT,
                      :ARTIMG1,
                      :ARTIMG2,
                      :ARTIMG3,
                      :ARTIMG4,
                      :ARTIMG5,
                      :ARTIMG6,
                      :ARTFEC,
                      :ARTFEC1
    END-EXEC.

    IF SQLSTATE = "02000"
       GO FIN-CARGAR
    END-IF.

    MOVE ARTCOD  TO PCOD.
    MOVE ARTDES  TO PDES.
    MOVE ARTFAM  TO PFAM FAMCOD.

    READ FAMILIAS NO LOCK
         INVALID KEY
                 MOVE " No existe ..." TO FAMDES
    END-READ.

    MOVE FAMDES  TO PFAMNOM.
    MOVE ARTOBS  TO POBS.
    MOVE ARTPRE1 TO PPRE1.
    MOVE ARTPRE2 TO PPRE2.
    MOVE ARTPRE3 TO PPRE3.
    MOVE ARTACT  TO PACT.

         MOVE POW-FALSE TO bCANCELO.
           MOVE "FLisArt1" TO PRT-FORMAT.
           MOVE "CUE001"  TO PRT-GROUP.
           WRITE FLisArt1.

           IF PRT-PROCE-CAN
                 MOVE POW-TRUE TO bCANCELO
                 CLOSE PRINT-FILE
                 MOVE 0 TO "MousePointer" OF POW-SELF
                 INVOKE BTLISTAR "SetFocus"
                 EXIT PROGRAM
           END-IF.

           IF NOT PRT-TODO-OK
                  MOVE SPACES TO WTEXT
            ADD POW-DMOK POW-DMICONERROR GIVING ESTILO
                  String "Impresora devolvió error [ "
                       PRT-STS2
                       " ] CUE001"
                       delimited by size INTO WTEXT
                END-STRING
                  INVOKE POW-SELF "DisplayMessage"
                         USING WTEXT
```

```
                                "Error"
                                ESTILO
                    END-INVOKE
                    MOVE 0 TO "MousePointer" OF POW-SELF
                  INVOKE BTLISTAR "SetFocus"
                    EXIT PROGRAM
                END-IF.

      ADD 1 TO LIN.

      IF LIN > nTOTLIN
         ADD 1 TO PAG
         PERFORM CABECERA THRU FIN-CABECERA
      END-IF.

 FIN-CARGAR.
      EXIT.
*
 CABECERA.
      MOVE POW-FALSE TO bCANCELO.

      MOVE PAG TO PPAG.


              *> Imprimo las lineas de la cabecera
              MOVE "FLisArt1" TO PRT-FORMAT.
              MOVE "CAB001"   TO PRT-GROUP.
              WRITE FLisArt1.

              IF PRT-PROCE-CAN
                    MOVE POW-TRUE TO bCANCELO
                    CLOSE PRINT-FILE
                    GO FIN-CABECERA
              END-IF.

              IF NOT PRT-TODO-OK
                     MOVE SPACES TO WTEXT
                     MOVE 0 TO ESTILO
               ADD POW-DMOK POW-DMICONERROR GIVING ESTILO
                     String "Impresora devolvió error [ "
                          PRT-STS2
                          " ] CAB001"
                          delimited by size INTO WTEXT
                  END-STRING
                    INVOKE POW-SELF "DisplayMessage"
                           USING WTEXT
                                 "Error"
                                 ESTILO
                    END-INVOKE
                    EXIT PROGRAM
                END-IF.
           MOVE 8 TO LIN.
 FIN-CABECERA.
      EXIT.
*
 MONTAR-CONDICIONES.
      MOVE SPACES TO WTEXT.
      COMPUTE C = FUNCTION STORED-CHAR-LENGTH (WCOD10).
      COMPUTE D = FUNCTION STORED-CHAR-LENGTH (WCOD20).

      STRING "Condiciones: Desde  "
             WCOD10 (1:C)
             "/"
             WCOD30
             "  Hasta  "
             WCOD20 (1:D)
             "/"
             WCOD40
```

```
            "." DELIMITED BY SIZE INTO WTEXT
      END-STRING.
```

Subrutina “VACIAR”

```
 ENVIRONMENT     DIVISION.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.
*
      MOVE SPACES TO "Text"    OF CAMPO1.
      MOVE SPACES TO "Text"    OF CAMPO2.
      MOVE SPACES TO "Text"    OF CAMPO3.
      MOVE SPACES TO "Text"    OF CAMPO4.

      MOVE SPACES TO "Caption" OF INFORMACION1.
      MOVE SPACES TO "Caption" OF INFORMACION2.
      MOVE SPACES TO "Caption" OF INFORMACION3.
      MOVE SPACES TO "Caption" OF INFORMACION4.
```

Consideraciones:

PWC, incluye una serie de subrutinas “especiales”, con las que podemos hacer bastantes cosas, como por ejemplo, comprobar si existe un fichero, una carpeta, borrar carpetas, mover ficheros, etc. Todas estas subrutinas, empiezan por el prefijo ***CBL_nombre_de_subrutina***, y su estructura depende de la función que queramos ejecutar, pero lo que **sí es imprescindible** para poder usarlas dentro de nuestros programas, es incluir la librería **F3BICBLR.lib**, que se encuentra en la carpeta ***C:\Program Files (x86)\Fujitsu NetCOBOL for Windows***, en el manual de PWC, vienen todas las rutinas CBL bastante bien explicadas, además de traer un ejemplo de uso.

Rutinas CBL_xxxxx/CBL_xxxxx2/CBL_64BIT_xxxx:

Bueno, pues Begoguay, del foro www.cobolforo.es, me ha pedido si puedo poner algunos ejemplos de las rutinas CBL más utilizadas de las que trae PWC incluidas, sobre todo por tener los ejemplos en español, en vez de en inglés, y evitar en lo posible los fallos al traducir.

La mayoría de estas rutinas, devuelven un valor de resultado de la operación en una variable numérica, que en el manual, suele definirse como `01 status-code PIC S9(4) COMP-5`. El nombre, le podemos poner el que queramos, status-code o retorno, o lo que sea, lo que sí que deberíamos de respetar es la `PICTURE` y el tipo de dato (`COMP-5`), porque, aunque no sea obligatorio, con ciertas rutinas, si no coincide bien el formato, el valor de retorno, no es correcto.

En las rutinas de manejo de ficheros, existen en la mayoría de ellas 2 versiones de la rutina, que se diferencian porque una de ellas, la más moderna, se le incluye un **2** al final del nombre, (CBL_CREATE_FILE y CBL_CREATE_FILE2 o CBL_OPEN_FILE y CBL_OPEN_FILE2), la diferencias comunes entre unas y otras son:

- La que no termina en **2**, pertenecen a la versión 3 de PWC, y en ésa versión no se pueden ejecutar las nuevas.
- La que termina en **2**, salvo contados casos, todos los parámetros son los mismos que las que no lo llevan, pero en las antiguas, el nombre de fichero, (incluida la ruta del fichero), no soporta espacios. Por lo que cuando sea iguales, en todos los parámetros, no voy a explicar las 2 rutinas, si no la más nueva, (la que termina en **2**).

Y a partir de la versión 7 de PWC, (creo que es de la 7), existen también rutinas para el manejo de ficheros de 64Bits, aunque PWC, no genera ejecutables de 64Bits, (ni siquiera en la versión 11, que supuestamente la sacaron para tener compatibilidad con Windows 10 x64), por lo que, como se suele decir, “mandagüevos” **:-(**. Los parámetros para la versión **2** y la de **64Bits**, son los mismos. Aunque hay que decir que, creo que las rutinas de 64Bits, están puestas exclusivamente para Solaris ®, y que si usamos Linux o Solaris, éstos no soportan el status-code de retorno de la llamada.

En la mayoría de los casos, en las rutinas de manejo de ficheros, el valor de la variable de retorno, si no es 0 (cero), el valor es el mismo que el file-status de los ficheros,(90, 35, 22, etc.). Por otro lado, cuando usemos la rutina de abrir o crear un fichero, hay un parámetro que es ***file-handle***, que vendría a ser el “manejador” del fichero, para estas rutinas, al llamarlas, su valor ha de ser espacios, y cuando ejecutamos la llamada, PWC, nos devuelve en esa variable, un determinado valor, que será único para esa acción, si más adelante, vamos a cerrar el fichero, (CBL_CLOSE_FILE) o descargar datos en un fichero (CBL_FLUSH_FILE), no haremos referencia al fichero por el nombre, si no por el “manejador”, por ello, todas las rutinas CBL, que hacen referencia al “manejador” en vez de al nombre del fichero directamente, no tienen una versión **2** de la rutina, (no existe CBL_CLOSE_FILE2 o CBL_FLUSH_FILE2) y al compilar, nos dará un error más o menos como el siguiente:

MainForm.obj : error LNK2001: unresolved external symbol _CBL_CLOSE_FILE2

CBL_OPEN_FILE/CBL_OPEN_FILE2/CBL_OPEN_64BIT_FILE

Abre un fichero

```
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  FICHERO                 PIC X(255).
 01  ACCESO                  PIC 9(4) BINARY.
 01  EXCLUSION               PIC 9(4) BINARY.
 01  FUTURO                  PIC 9(4) BINARY VALUE 0.
 01  MANEJADOR               PIC X(4).
 01  ESTADO                  PIC S9(4) COMP-5.
*
 01  I                       PIC 999 BINARY.
 PROCEDURE       DIVISION.
*
* FICHERO = Ruta y nombre del fichero, debe terminar en un espacio en blanco o null
* ACCESO = Modo de acceso al fichero: (El valor normal sería 3)
*                 1 - Sólo Lectura
*                 2 - Sólo Escritura
*                 3 - Lectura/Escritura
* EXCLUSION = Especifica uno de los modos de exclusión siguiente: (El valor normal
sería 3)
*                 0 - No permite leer o escribir
*                 1 - No permite escribir
*                 2 - No permite leer
*                 3 - Sin restricciones
* FUTURO = Está pensado para futuras espansiones y debe valer 0 (cero)
* MANEJADOR = Es para el "handled" del dichero, (uso interno) y debe vale espacios
* ESTADO = Código de retorno de la función:
*                 0  - Sin error
*                 -1 - Algún error
*
*
 INICIO.
     MOVE SPACES TO FICHERO.
     MOVE "C:\Articulos\Datos\PRUEBA.TXT" & X"00" TO FICHERO.
     MOVE 3 TO EXCLUSION ACCESO.
     CALL "CBL_OPEN_FILE2"
          USING FICHERO
                ACCESO
                EXCLUSION
                FUTURO
                MANEJADOR
          RETURNING ESTADO
     END-CALL.
*
     IF ESTADO NOT = 0
        INVOKE POW-SELF "DisplayMessage"
               USING "Error en la llamada a la rutina CBL ..."
                     ESTADO
                     POW-DMICONERROR
        END-INVOKE
     END-IF.
*
```

CBL_CREATE_FILE/CBL_CREATE_FILE2/CBL_CREATE_64BIT_FILE

Crea un fichero

```
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  FICHERO                 PIC X(255).
 01  ACCESO                  PIC 9(4) BINARY.
 01  EXCLUSION               PIC 9(4) BINARY.
 01  FUTURO                  PIC 9(4) BINARY VALUE 0.
 01  MANEJADOR               PIC X(4).
```

```
 01  ESTADO                  PIC S9(4) COMP-5.
*
 01  I                       PIC 999 BINARY.
 PROCEDURE       DIVISION.
*
* FICHERO = Ruta y nombre del fichero, debe terminar en un espacio en blanco o null
* ACCESO = Modo de acceso al fichero: (El valor normal sería 3)
*                 1 - Sólo Lectura
*                 2 - Sólo Escritura
*                 3 - Lectura/Escritura
* EXCLUSION = Especifica uno de los modos de exclusión siguiente: (El valor normal
sería 3)
*                 0 - No permite leer o escribir
*                 1 - No permite escribir
*                 2 - No permite leer
*                 3 - Sin restricciones
* FUTURO = Está pensado para futuras espansiones y debe valer 0 (cero)
* MANEJADOR = Es para el "handled" del dichero, (uso interno) y debe vale espacios
* ESTADO = Código de retorno de la función:
*                 0  - Sin error
*                 -1 - Algún error
*
*
 INICIO.
     MOVE SPACES TO FICHERO.
     MOVE "C:\Articulos\Datos\PRUEBA.TXT" & X"00" TO FICHERO.
     MOVE 3 TO EXCLUSION ACCESO.
     CALL "CBL_CREATE_FILE2"
          USING FICHERO
                ACCESO
                EXCLUSION
                FUTURO
                MANEJADOR
          RETURNING ESTADO
     END-CALL.
*
     IF ESTADO NOT = 0
        INVOKE POW-SELF "DisplayMessage"
               USING "Error en la llamada a la rutina CBL ..."
                     ESTADO
                     POW-DMICONERROR
        END-INVOKE
     END-IF.
*
```

CBL_CLOSE_FILE/CBL_CLOSE_64BIT_FILE

Cierra un Fichero.

```
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  FICHERO                 PIC X(255).
 01  ACCESO                  PIC 9(4) BINARY.
 01  EXCLUSION               PIC 9(4) BINARY.
 01  FUTURO                  PIC 9(4) BINARY VALUE 0.
 01  MANEJADOR               PIC X(4).
 01  ESTADO                  PIC S9(4) COMP-5.
*
 01  I                       PIC 999 BINARY.
 PROCEDURE       DIVISION.
*
* FICHERO = Ruta y nombre del fichero, debe terminar en un espacio en blanco o null
* ACCESO = Modo de acceso al fichero: (El valor normal sería 3)
*                 1 - Sólo Lectura
*                 2 - Sólo Escritura
*                 3 - Lectura/Escritura
* EXCLUSION = Especifica uno de los modos de exclusión siguiente: (El valor normal
sería 3)
```

```
*                0 - No permite leer o escribir
*                1 - No permite escribir
*                2 - No permite leer
*                3 - Sin restricciones
* FUTURO = Está pensado para futuras espansiones y debe valer 0 (cero)
* MANEJADOR = Es para el "handled" del dichero, (uso interno) y debe vale espacios
* ESTADO = Código de retorno de la función:
*                0  - Sin error
*                -1 - Algún error
*
*
 INICIO.
     MOVE SPACES TO FICHERO.
     MOVE "C:\Articulos\Datos\PRUEBA.TXT" & X"00" TO FICHERO.
     MOVE 3 TO EXCLUSION ACCESO.
     CALL "CBL_OPEN_FILE2"
          USING FICHERO
                ACCESO
                EXCLUSION
                FUTURO
                MANEJADOR
          RETURNING ESTADO
     END-CALL.
*
     IF ESTADO NOT = 0
        INVOKE POW-SELF "DisplayMessage"
               USING "Error en la llamada a la rutina CBL (Open)..."
                     ESTADO
                     POW-DMICONERROR
        END-INVOKE
        INVOKE POW-SELF "CloseForm"
        EXIT PROGRAM
     END-IF.
*
     CALL "CBL_CLOSE_FILE"
          USING MANEJADOR
          RETURNING ESTADO
     END-CALL.
*
     IF ESTADO NOT = 0
        INVOKE POW-SELF "DisplayMessage"
               USING "Error en la llamada a la rutina CBL (Close)..."
                     ESTADO
                     POW-DMICONERROR
        END-INVOKE
     END-IF.
*
```

CBL_FLUSH_FILE/CBL_FLUSH_64BIT_FILE

“Descarga” un Fichero, nunca la he usado, puesto que no me ha hecho falta grabar ficheros con estas rutinas, por lo que no tengo muy claro como funciona, ni ésta rutina, ni la de grabar o leer, así que pongo lo poco que sé, y si alguien que la haya usado, quiere enviarme ejemplos detallados, pues se agradecerá.

```
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  FICHERO                 PIC X(255).
 01  ACCESO                  PIC 9(4) BINARY.
 01  EXCLUSION               PIC 9(4) BINARY.
 01  FUTURO                  PIC 9(4) BINARY VALUE 0.
 01  MANEJADOR               PIC X(4).
 01  ESTADO                  PIC S9(4) COMP-5.
*
 01  I                       PIC 999 BINARY.
 PROCEDURE       DIVISION.
*
* FICHERO = Ruta y nombre del fichero, debe terminar en un espacio en blanco o null
* ACCESO = Modo de acceso al fichero: (El valor normal sería 3)
```

```
*                 1 - Sólo Lectura
*                 2 - Sólo Escritura
*                 3 - Lectura/Escritura
* EXCLUSION = Especifica uno de los modos de exclusión siguiente: (El valor normal
sería 3)
*                 0 - No permite leer o escribir
*                 1 - No permite escribir
*                 2 - No permite leer
*                 3 - Sin restricciones
* FUTURO = Está pensado para futuras espansiones y debe valer 0 (cero)
* MANEJADOR = Es para el "handled" del dichero, (uso interno) y debe vale espacios
* ESTADO = Código de retorno de la función:
*                 0  - Sin error
*                 -1 - Algún error
*
*
 INICIO.
     MOVE SPACES TO FICHERO.
     MOVE "C:\Articulos\Datos\PRUEBA.TXT" & X"00" TO FICHERO.
     MOVE 3 TO EXCLUSION ACCESO.
     CALL "CBL_OPEN_FILE2"
          USING FICHERO
                ACCESO
                EXCLUSION
                FUTURO
                MANEJADOR
          RETURNING ESTADO
     END-CALL.
*
     IF ESTADO NOT = 0
        INVOKE POW-SELF "DisplayMessage"
               USING "Error en la llamada a la rutina CBL (Open)..."
                     ESTADO
                     POW-DMICONERROR
        END-INVOKE
        INVOKE POW-SELF "CloseForm"
        EXIT PROGRAM
     END-IF.
*
     CALL "CBL_FLUSH_FILE"
          USING MANEJADOR
          RETURNING ESTADO
     END-CALL.
*
     IF ESTADO NOT = 0
        INVOKE POW-SELF "DisplayMessage"
               USING "Error en la llamada a la rutina CBL (Flush)..."
                     ESTADO
                     POW-DMICONERROR
        END-INVOKE
     END-IF.
*
```

CBL_CHANGE_DIR/CBL_CHANGE_DIR2

Cambia el directorio (carpeta) actual, por el especificado. Cuando ejecutamos un programa hecho en PWC, el directorio o carpeta por defecto, es en el que se encuentra el ejecutable que hemos iniciado y, a no ser que lo cambiemos desde el programa a través de alguna instrucción tipo a `INVOKE POW-SELF "GetFileName" USING ...`, se mantendrá invariablemente en esa carpeta, con lo que si creamos algún archivo con la instrucción *CBL_CREATE_FILE2*, por ejemplo, se creará en la carpeta del ejecutable.

```
DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  RUTA                    PIC X(255).
 01  ESTADO                  PIC S9(4) COMP-5.
 PROCEDURE       DIVISION.
*
```

```
* RUTA = Nueva carpeta o ruta. debe terminar en un carácter en blanco o null
* ESTADO = Código de retorno de la función:
*                  0  - Sin error
*                 -1  - Algún error
*
 INICIO.
     MOVE SPACES TO RUTA.
     MOVE "C:\Articulos\Datos" & X"00" TO RUTA.
     CALL "CBL_CHANGE_DIR2"
          USING RUTA
          RETURNING ESTADO
     END-CALL.
*
     IF ESTADO NOT = 0
        INVOKE POW-SELF "DisplayMessage"
               USING "Error en la llamada a la rutina CBL..."
                     ESTADO
                     POW-DMICONERROR
        END-INVOKE
        INVOKE POW-SELF "CloseForm"
        EXIT PROGRAM
     END-IF.
*
```

CBL_CHECK_FILE_EXIST/CBL_CHECK_FILE_EXIST2

Comprueba si existe o no un fichero., además, devuelve tamaño, fecha y hora de creación del fichero.

```
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  FICHERO                  PIC X(255).
 01  DETALLES.
     02  TAMANO               PIC 9(18) BINARY.
     02  FECHA.
         03  DIA              PIC 9(4) BINARY.
         03  MES              PIC 9(4) BINARY.
         03  ANNO             PIC 9(4) BINARY.
     02  HORA.
         03  HH               PIC 9(4) BINARY.
         03  MM               PIC 9(4) BINARY.
         03  SS               PIC 9(4) BINARY.
         03  CC               PIC 9(4) BINARY.
 01  ESTADO                   PIC S9(4) COMP-5.
 PROCEDURE       DIVISION.
*
* FICHERO = Nombre del fichero a comprobar, debe terminar en un carácter en blanco o
null
* DETALLES = Detalles del fichero
*          TAMANO = Tamaño del fichero (en bytes)
*          FECHA/HORA = Fecha y Hora de creación del fichero
* ESTADO = Código de retorno de la función:
*                  0  - Sin error
*                 -1  - Algún error
*
 INICIO.
     MOVE SPACES TO FICHERO.
     MOVE "C:\Articulos\Datos\PRUEBA.TXT" & X"00" TO FICHERO.
     CALL "CBL_CHECK_FILE_EXIST2"
          USING FICHERO
                DETALLES
          RETURNING ESTADO
     END-CALL.
*
     IF ESTADO NOT = 0
        INVOKE POW-SELF "DisplayMessage"
               USING "Error en la llamada a la rutina CBL..."
                     ESTADO
                     POW-DMICONERROR
```

```
        END-INVOKE
        INVOKE POW-SELF "CloseForm"
        EXIT PROGRAM
     END-IF.
*
```

CBL_COPY_FILE/CBL_COPY_FILE2

Copia un fichero en otro. Si no especificamos una ruta, se usará la que haya por defecto en ese momento. ¡ atención !, si el fichero destino existe, lo sobreescribirá con el nuevo sin dar ningún error o advertencia.

```
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  FICHERO                 PIC X(255).
 01  FICHERO2                PIC X(255).
 01  ESTADO                  PIC S9(4) COMP-5.
 PROCEDURE       DIVISION.
*
* FICHERO = FICHERO de ORIGEN. debe terminar en un carácter en blanco o null
* FICHERO2 = FICHERO de DESTINO. debe terminar en un carácter en blanco o null
* ESTADO = Código de retorno de la función:
*                  0  - Sin error
*                 -1  - Algún error
*
 INICIO.
     MOVE SPACES TO FICHERO FICHERO2.
     MOVE "C:\Articulos\Datos\PRUEBA.TXT" & X"00" TO FICHERO.
     MOVE "C:\Articulos\Datos\Copia PRUEBA.TXT" & X"00" TO FICHERO2.
     CALL "CBL_COPY_FILE2"
          USING FICHERO
                FICHERO2
          RETURNING ESTADO
     END-CALL.
*
     IF ESTADO NOT = 0
        INVOKE POW-SELF "DisplayMessage"
               USING "Error en la llamada a la rutina CBL..."
                     ESTADO
                     POW-DMICONERROR
        END-INVOKE
        INVOKE POW-SELF "CloseForm"
        EXIT PROGRAM
     END-IF.
*
```

CBL_CREATE_DIR/CBL_CREATE_DIR2

Crea una carpeta o directorio en la ruta especificada. Si la carpeta ya existe, nos devolverá error (-1).

```
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  RUTA                    PIC X(255).
 01  ESTADO                  PIC S9(4) COMP-5.
 PROCEDURE       DIVISION.
*
* RUTA = RUTA de la CARPETA a crear. debe terminar en un carácter en blanco o null
* ESTADO = Código de retorno de la función:
*                  0  - Sin error
*                 -1  - Algún error
*
 INICIO.
     MOVE SPACES TO RUTA.
     MOVE "C:\Articulos\Datos\PRUEBA\" & X"00" TO RUTA.
     CALL "CBL_CREATE_DIR2"
          USING RUTA
          RETURNING ESTADO
     END-CALL.
```

```
*
     IF ESTADO NOT = 0
        INVOKE POW-SELF "DisplayMessage"
               USING "Error en la llamada a la rutina CBL..."
                     ESTADO
                     POW-DMICONERROR
        END-INVOKE
        INVOKE POW-SELF "CloseForm"
        EXIT PROGRAM
     END-IF.
*
```

CBL_DELETE_DIR/CBL_DELETE_DIR2

Borra el directorio o carpeta especificado. Si no se especifica ninguno, borrará la carpeta actual. Si no se puede borrar por alguna razón, (no existe la carpeta o no está vacía), nos dará error. (-1).

```
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  RUTA                    PIC X(255).
 01  ESTADO                  PIC S9(4) COMP-5.
 PROCEDURE       DIVISION.
*
* RUTA = RUTA de la CARPETA a crear. debe terminar en un carácter en blanco o null
* ESTADO = Código de retorno de la función:
*                  0  - Sin error
*                 -1  - Algún error
*
 INICIO.
     MOVE SPACES TO RUTA.
     MOVE "C:\Articulos\Datos\PRUEBA\" & X"00" TO RUTA.
     CALL "CBL_DELETE_DIR2"
          USING RUTA
          RETURNING ESTADO
     END-CALL.
*
     IF ESTADO NOT = 0
        INVOKE POW-SELF "DisplayMessage"
               USING "Error en la llamada a la rutina CBL..."
                     ESTADO
                     POW-DMICONERROR
        END-INVOKE
        INVOKE POW-SELF "CloseForm"
        EXIT PROGRAM
     END-IF.
*
```

CBL_DELETE_FILE/CBL_DELETE_FILE2

Borra un fichero especificado.

```
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  FICHERO                 PIC X(255).
 01  ESTADO                  PIC S9(4) COMP-5.
 PROCEDURE       DIVISION.
*
* FICHERO = FICHERO a borrar. debe terminar en un carácter en blanco o null
* ESTADO = Código de retorno de la función:
*                  0  - Sin error
*                 -1  - Algún error
*
 INICIO.
     MOVE SPACES TO FICHERO.
     MOVE "C:\Articulos\Datos\PRUEBA.TXT" & X"00" TO FICHERO.
     CALL "CBL_DELETE_FILE2"
          USING FICHERO
```

```
          RETURNING ESTADO
     END-CALL.
*
     IF ESTADO NOT = 0
        INVOKE POW-SELF "DisplayMessage"
               USING "Error en la llamada a la rutina CBL..."
                     ESTADO
                     POW-DMICONERROR
        END-INVOKE
        INVOKE POW-SELF "CloseForm"
        EXIT PROGRAM
     END-IF.
*
```

CBL_READ_DIR

Devuelve el nombre de la carpeta/ruta o directorio activo. (En el que nos encontramos en ese momento).

```
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  RUTA                    PIC X(255).
 01  TAMANO                  PIC 9(4) BINARY.
 01  ESTADO                  PIC S9(4) COMP-5.
 PROCEDURE       DIVISION.
*
* RUTA = RUTA a LEER
* TAMANO = TAMAÑO del área donde se almacena el nombre de la ruta
* ESTADO = Código de retorno de la función:
*                 0  - Sin error
*                -1  - Algún error
*
 INICIO.
*    MOVE SPACES TO RUTA.
*    MOVE "C:\Articulos\Datos\" & X"00" TO RUTA.
     CALL "CBL_READ_DIR"
          USING RUTA
                TAMANO
          RETURNING ESTADO
     END-CALL.
*
     IF ESTADO NOT = 0
        INVOKE POW-SELF "DisplayMessage"
               USING "Error en la llamada a la rutina CBL..."
                     ESTADO
                     POW-DMICONERROR
        END-INVOKE
        INVOKE POW-SELF "CloseForm"
        EXIT PROGRAM
     END-IF.
*
```

CBL_RENAME_FILE/CBL_RENAME_FILE2

Cambia el nombre de un fichero. Si el nuevo nombre ya existe, nos devolverá error. (-1).

```
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  FICHERO                 PIC X(255).
 01  FICHERO2                PIC X(255).
 01  ESTADO                  PIC S9(4) COMP-5.
 PROCEDURE       DIVISION.
*
* FICHERO = Nombre antiguo del fichero
* FICHERO2 = Nombre nuevo del fichero
* ESTADO = Código de retorno de la función:
*                 0  - Sin error
*                -1  - Algún error
```

```
*
 INICIO.
     MOVE SPACES TO FICHERO FICHERO2
     MOVE "C:\Articulos\Datos\PRUEBA.TXT" & X"00" TO FICHERO.
     MOVE "C:\Articulos\Datos\Nueva PRUEBA.TXT" & X"00" TO FICHERO2.
     CALL "CBL_RENAME_FILE2"
          USING FICHERO
                FICHERO2
          RETURNING ESTADO
     END-CALL.
*
     IF ESTADO NOT = 0
        INVOKE POW-SELF "DisplayMessage"
               USING "Error en la llamada a la rutina CBL..."
                     ESTADO
                     POW-DMICONERROR
        END-INVOKE
        INVOKE POW-SELF "CloseForm"
        EXIT PROGRAM
     END-IF.
*
```

CBL_TOUPPER/CBL_TOLOWER

Cambia de Minúsculas a Mayúsculas (CBL_TOUPPER) y viceversa (CBL_TOLOWER).

```
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  CADENA                  PIC X(8192).
 01  LEN                     PIC 9(4)  COMP-5.
 01  ESTADO                  PIC S9(4) COMP-5.
 PROCEDURE       DIVISION.
*
* CADENA = Cadena a convertir
* LEN = Longitud de la cadena a convertir
* ESTADO = Código de retorno de la función:
*                  0 - Sin error
*                 -1 - Algún error
*
 INICIO.
*
     MOVE "convertir de minúsculas a mayúsculas ..." TO CADENA.
     COMPUTE LEN = FUNCTION STORED-CHAR-LENGTH (CADENA).
     CALL "CBL_TOUPPER"
          USING CADENA
                BY VALUE LEN
          RETURNING ESTADO
     END-CALL.
*
     IF ESTADO NOT = 0
        INVOKE POW-SELF "DisplayMessage"
               USING "Error en la llamada a la rutina CBL..."
                     ESTADO
                     POW-DMICONERROR
        END-INVOKE
        INVOKE POW-SELF "CloseForm"
        EXIT PROGRAM
     END-IF.
*
*-------------------------------------------------------------------------
*
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 01  CADENA              PIC X(8192).
 01  LEN                       PIC 9(4)  COMP-5.
 01  ESTADO              PIC S9(4) COMP-5.
 PROCEDURE       DIVISION.
*
```

```
* CADENA = Cadena a convertir
* LEN = Longitud de la cadena a convertir
* ESTADO = Código de retorno de la función:
*                  0 - Sin error
*                 -1 - Algún error
*
 INICIO.
*
     MOVE "CONVERTIR DE MAYÚSCULAS A MINÚSCULAS..." TO CADENA.
     COMPUTE LEN = FUNCTION STORED-CHAR-LENGTH (CADENA).
     CALL "CBL_TOLOWER"
          USING CADENA
                BY VALUE LEN
          RETURNING ESTADO
     END-CALL.
*
     IF ESTADO NOT = 0
        INVOKE POW-SELF "DisplayMessage"
               USING "Error en la llamada a la rutina CBL..."
                     ESTADO
                     POW-DMICONERROR
        END-INVOKE
        INVOKE POW-SELF "CloseForm"
        EXIT PROGRAM
     END-IF.
*
```

CBL_ALARM_SOUND

Genera el sonido de alarma de Windows. El equivalente a `INVOKE POW-SELF "Alarm" USING POW-MBEXCLAMATION.`

```
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE       DIVISION.
*
 INICIO.
*
     CALL "CBL_ALARM_SOUND".
*
```

Otras consideraciones.

Existen muchas más rutinas CBL, como son para un **y** ó un **o lógicos, xor, eq**, etc, rutinas para el teclado, pantalla y ratón, aunque éstas últimas sólo para modo consola, pero como no son de uso muy cotidiano, no las he puesto, si a alguien le interesan, que lo comente y las pondría.

FUNCIONES INTRÍNSECAS.

PWC, provee de una cantidad de funciones intrínsecas, que son ni más ni menos, que funciones que realizan rutinas más o menos complicadas, como calcular diferencias entre fechas, el arcoseno, la tangente, mayúsculas a minúsculas, etc. Todas las funciones, van precedidas de la palabra reservada **FUNCTION**, por ejemplo, … **FUNCTION STORED-CHAR-LENGTH (APELLIDOS).** Y dependiendo del tipo de función que vayamos a utilizar, la mayoría han de comenzar con COMPUTE o con MOVE, dependiendo si el valor que nos ha de devolver la función es numérico, (por ejemplo el arcotangente), o una cadena de caracteres, (por ejemplo la fecha del día).

(F. 1)

```
COMPUTE LEN = FUNCTION STORED-CHAR-LENGTH (APELLIDOS).
```

(F. 2)

```
PERFORM VARYING LEN FROM 50 BY -1
        UNTIL LEN < 1
              OR APELLIDOS (LEN:1) = SPACES
                 CONTINUE
END-PERFORM.
```

En la forma 1, os muestro cómo se haría con una función intrínseca, para saber la cantidad de caracteres usados que tiene la variable APELLIDOS. Donde si no tuviéramos ésta función, nos tocaría hacerlo manualmente con unas cuantas lineas más, (como se ha hecho desde siempre), tal y como muestro en la 2ª forma

(F. 1)

```
MOVE FUNCTION CURRENT-DATE TO FECHA-HORA.
```

(F. 2)

```
ACCEPT FECHA FROM DATE.
ACCEPT HORA FROM TIME.
ADD 2000 TO AA.
```

Y en éste otro ejemplo, en la forma 1, mueve a la variable FECHA-HORA, la fecha y la hora del sistema con el año con 4 decimales. Y en la forma 2,

Y como hay bastantes funciones, y algunas de ellas, son muy útiles, vamos a explicar las más utilizadas.

FUNCIÓN ADDR

Aunque no creo que la uséis mucho, con ésta función obtenemos la dirección de memoria de un dato, (el equivalente a un puntero en C o a la antigua y olvidada instrucción POKE en BASIC), por lo que sólo podemos almacenar el valor en una variable del tipo POINTER. Se utiliza sobre todo para pasar datos entre programas COBOL y programas C.

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  PUNTERO                  USAGE IS POINTER.
 01  VARIABLE                 PIC X(50).
*
 PROCEDURE DIVISION.
*
     MOVE FUNCTION ADDR (VARIABLE) TO PUNTERO.
```

Si queremos usar la variable como puntero en vez de directamente, algo no muy lógico en COBOL, lo podemos hacer de la siguiente manera:

```
     MOVE SPACES TO PUNTERO->VARIABLE.
```

ésto es exactamente igual que escribir:

```
     MOVE SPACES TO VARIABLE.
```

FUNCIÓN LENG/LENGTH

Devuelve el tamaño **total** de una variable. (El tamaño con que definimos la variable en la WORKING-STORAGE), la diferencia entre una y otra función es que, la primera, devuelve el tamaño en bytes, y la otra, devuelve el tamaño en “posiciones”. Tener en cuenta que, como PWC soporta DBCS, (**D**ouble **B**yte **C**haracter **S**et), que es el juego de caracteres de doble byte, (lo que serían caracteres chinos, japoneses, etc), en caso de que estuviésemos usando DBCS, si la función Length nos diese 10 caracteres, por ejemplo, la función, nos daría 20.

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  LON                      PIC 99.
 01  LON1                     PIC 99.
 01  VARIABLE                 PIC X(50).
*
 PROCEDURE DIVISION.
*
     MOVE  "Hola Mundo" TO VARIABLE.
     COMPUTE LON  = FUNCTION LENG (VARIABLE).
     COMPUTE LON1 = FUNCTION LENG (VARIABLE).
```

En ambos casos, nos devolvería 50, tanto en LON, como en LON1, porque es el tamaño con el que hemos definido VARIABLE en la WORKING-STORAGE, dando lo mismo la instrucción `MOVE "Hola ..."`.

FUNCIÓN STORED-CHAR-LENGTH

Devuelve el tamaño que tiene ocupado una variable alfanumérica de su total.

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  LON                      PIC 99.
 01  VARIABLE                 PIC X(50).
 01  LON1                     PIC 99.
 01  VARIABLE1                PIC X(50).
*
```

```
 PROCEDURE DIVISION.
*
     MOVE "Hola Mundo" TO VARIABLE.
     COMPUTE LON  = FUNCTION STORED-CHAR-LENGTH (VARIABLE).
     MOVE "    Hola Mundo    " TO VARIABLE1.
     COMPUTE LON1 = FUNDTION STORED-CHAR-LENGTH (VARIABLE1).
```

En éste ejemplo, la variable LON, pasará a vale 10 y la variable LON1, pasará a valer 14, puesto que los espacios en blanco de la izquierda, **sí** se cuentan, pero los espacios de la derecha no, equivaldría a la instrucción LTRIM de BASIC o de MySQL.

FUNCIÓN CURRENT-DATE

Devuelve la fecha y hora actual del sistema, se diferencia de la función `ACCEPT … FROM DATE`, en que ésta última, devuelve del año los 2 últimos dígitos, mientras que la función, devuelve **siempre**, el año en formato cuatro dígitos.

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  AHORA.
     03  FECHA.
         05  AA               PIC XXXX.
         05  MM               PIC XX.
         05  DD               PIC XX.
     03  HORA.
         05  HH               PIC XX.
         05  MM               PIC XX.
         05  SS               PIC XX.
         05  DC               PIC XX.
     03  LOCAL-H.
         05  DIF              PIC X.      *> Zona Horaria
         05  DIF-H            PIC XX.     *> Diferencia en horas con Greenwich
         05  DIF-M            PIC XX.     *> Diferencia en minutos.
*
 01  FECHA1.
     03  DD                   PIC 99.
     03                       PIC X VALUE "/".
     03  MM                   PIC 99.
     03                       PIC X VALUE "/".
     03  AA                   PIC 9999.
 01  HORA1.
     03  HH                   PIC 99.
     03                       PIC X VALUE ":".
     03  MM                   PIC 99.
     03                       PIC X VALUE ":".
     03  SS                   PIC 99.
     03                       PIC X VALUE "'".
     03  DC                   PIC 99.
 PROCEDURE DIVISION.
*
     MOVE FUNCTION CURRENT-DATE TO AHORA.
*
     MOVE CORRESPONDING FECHA TO FECHA1.
     MOVE CORR          HORA  TO HORA1.
```

Yo suelo definir la variable AHORA, hasta `03  LOCAL-H.` porque nunca he utilizado las diferencias horarias, con lo que al no definirlos, esos datos se "pierden en el limbo", con esto quiero decir, que no es necesario definir todos los datos, si no sólo los que vamos a utilizar, pero respetar el orden, no quitéis la fecha y dejéis la hora, porque no es así, el orden hay que respetarlo.

FUNCIÓN SUM

Devuelve la SUMA de los valores o variables que le hayamos pasado como argumentos.

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  UNO                 PIC 9 VALUE 1.
 01  DOS                 PIC 9 VALUE 2.
 01  TRES                PIC 9 VALUE 3.
 01  CUATRO              PIC 9 VALUE 4.
*
 01  SUMA                PIC 99.
 01  SUMA1               PIC 999.
*
 01  NUMERO.
     03  NUM             PIC 99 OCCURS 9.
*
 01  I                   PIC 99 BINARY.
*
 PROCEDURE DIVISION.
*
     INITIALIZE NUMERO SUMA SUMA1.
     PERFORM VARYING I FROM 1 BY 1 UNTIL I > 9
             COMPUTE NUM (I) = I * 10
     END-PERFORM.
*
     COMPUTE SUMA  = FUNCTION SUM (UNO, DOS, TRES, CUATRO, 5, 6).
     COMPUTE SUMA1 = FUNCTION SUM (NUM (ALL)).
```

Con éste programa, la variable SUMA, tomará el valor 21 y la variable SUMA1, tomará el valor 450.

FUNCIÓN MEDIAN

Devuelve la MEDIANA de los valores o variables que le hayamos pasado como argumentos.

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  UNO                 PIC 9 VALUE 1.
 01  DOS                 PIC 9 VALUE 2.
 01  TRES                PIC 9 VALUE 3.
 01  CUATRO              PIC 9 VALUE 4.
*
 01  MEDI                PIC 99V99.
 01  MEDI1               PIC 999V99.
*
 01  NUMERO.
     03  NUM             PIC 99 OCCURS 9.
*
 01  I                   PIC 99 BINARY.
*
 PROCEDURE DIVISION.
*
     INITIALIZE NUMERO MEDI MEDI1.
     PERFORM VARYING I FROM 1 BY 1 UNTIL I > 9
             COMPUTE NUM (I) = I * 10
     END-PERFORM.
*
     COMPUTE MEDI  = FUNCTION MEDIAN (UNO, DOS, TRES, CUATRO, 5).
     COMPUTE MEDI1 = FUNCTION MEDIAN (NUM (ALL)).
```

Con éste programa, la variable MEDI, tomará el valor 3 y la variable MEDI1, tomará el valor 50,00.

FUNCIÓN MEAN/MIDRANGE

Devuelve la MEDIA ARITMÉTICA (MEAN) o la MEDIA ARITMÉTICA del valor más alto y más bajo (MIDRANGE) de los valores o variables que le hayamos pasado como argumentos.

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  UNO                PIC 9 VALUE 1.
 01  DOS                PIC 9 VALUE 2.
 01  TRES               PIC 9 VALUE 3.
 01  CUATRO             PIC 9 VALUE 4.
*
 01  MEDI               PIC 99V99.
 01  MEDI1              PIC 999V99.
*
 01  NUMERO.
     03  NUM            PIC 99 OCCURS 9.
*
 01  I                  PIC 99 BINARY.
*
 PROCEDURE DIVISION.
*
     INITIALIZE NUMERO MEDI MEDI1.
     PERFORM VARYING I FROM 1 BY 1 UNTIL I > 9
             COMPUTE NUM (I) = I * 10
     END-PERFORM.
*
     COMPUTE MEDI  = FUNCTION MEAN (UNO, DOS, TRES, CUATRO, 5, 6).
     COMPUTE MEDI1 = FUNCTION MEAN (NUM (ALL)).
```

Con éste programa, la variable MEDI, tomará el valor 3,5 y la variable MEDI1, tomará el valor 50,00.

FUNCIÓN REM

Devuelve el resto de una división. (Equivaldría a `REMAINDER` en la instrucción `DIVIDE` en COBOL)

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  TRES               PIC 9 VALUE 3.
 01  CUATRO             PIC 9 VALUE 4.
*
 01  I                  PIC 9.
*
 PROCEDURE DIVISION.
*
     MOVE 0 TO I.
*
     COMPUTE I  = FUNCTION REM (CUATRO TRES).
```

Con éste programa, la variable I, tomará el valor 1.

FUNCIÓN MAX/MIN

Devuelve la MEDIANA de los valores o variables que le hayamos pasado como argumentos.

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  UNO                PIC 9 VALUE 1.
 01  DOS                PIC 9 VALUE 2.
 01  TRES               PIC 9 VALUE 3.
 01  CUATRO             PIC 9 VALUE 4.
*
```

```
 01  MAYOR              PIC 99.
 01  MENOR              PIC 99.
*
 01  NUMERO.
     03  NUM            PIC 99 OCCURS 9.
*
 01  I                  PIC 99 BINARY.
*
 PROCEDURE DIVISION.
*
     INITIALIZE NUMERO MAYOR MENOR.
     PERFORM VARYING I FROM 1 BY 1 UNTIL I > 9
             COMPUTE NUM (I) = I * 10
     END-PERFORM.
*
     COMPUTE MAYOR = FUNCTION MAX (UNO, DOS, 6, TRES, 9, 0, CUATRO, 5).
     COMPUTE MENOR = FUNCTION MIN (NUM (ALL)).
```

Con éste programa, la variable MAYOR, tomará el valor 9 y la variable MENOR, tomará el valor 10.

FUNCIÓN INTEGER-OF-DATE/DATE-OF-INTEGER

INTEGER-OF-DATE, toma una fecha en formato AAAAMMDD y devuelve el número de días que han pasado desde el 1 de enero de 1601 hasta la fecha pasada como argumento. DATE-OF-INTEGER, coge un valor numérico pasado como argumento, se lo suma a 1 de enero de 1601 y nos devuelve una fecha en formato AAAAMMDD. La mayor utilidad de ambas funciones, es para el cálculo de diferencia de fechas y para obtener fechas, sobre todo a la hora de calcular vencimientos.

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  FECHA.
     03  FECHA1              PIC 9(8).
     03  FECHA2              PIC 9(8).
 01  DIAS                    PIC S9(9) BINARY.
*
 PROCEDURE DIVISION.
*
     MOVE 19800101 TO FECHA1.
     MOVE 19991231 TO FECHA2.
*
     COMPUTE DIAS = FUNCTION INTEGER-OF-DATE (FECHA2) -
                    FUNCTION INTEGER-OF-DATE (FECHA1).
*
     INITIALIZE FECHA2.
     COMPUTE FECHA2 = FUNCTION DATE-OF-INTEGER (
                      (FUNCTION INTEGER-OF-DATE (FECHA1) + DIAS)
                                                ).
*
     DISPLAY FECHA2.
```

En éste programa, la variable DIAS tomará el valor 7304, que son los días que han pasado desde el 01/01/1980 hasta el 31/12/1999. En la segunda parte del programa, cogemos de nuevo la primera fecha, le volvemos a sumar los 7304 días para que nos vuelva a dar la segunda fecha.

FUNCIÓN SIN/COS/TAN

Devuelve el SENO, COSENO y TANGENTE de un valor pasado como argumento de la función. PWC, devuelve el valor en RADIANES, no en GRADOS, no sé si habrá alguna manera de que lo haga en grados, de todas maneras, recordar que la equivalencia es 1·rad = 57,29578° ó 2·π·rad=360°.

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  SENO                    PIC 99V9999.
 01  COSENO                  PIC S9V9999.
 01  TANG                    PIC 99V9999.
 01  PI                      PIC 9V9(8) VALUE 3,14159265 COMP-5.
*
 PROCEDURE DIVISION.
*
     MOVE 0 TO SENO COSENO TANG.
*
     COMPUTE SENO   = FUNCTION SIN (180).
     COMPUTE COSENO = FUNCTION COS (PI).
     COMPUTE TANG   = FUNCTION TAN (10).
*
```

En éste programa, la variable SENO valdrá 0,8012, la varibale COSENO valdrá -1,00 y TANG 0,6484, siempre en radianes.

FUNCIÓN ASIN/ACOS/ATAN

Devuelve el ARCOSENO, ARCOCOSENO y ARCOTANGENTE de un valor pasado como argumento de la función. PWC, devuelve el valor en RADIANES, no en GRADOS, no sé si habrá alguna manera de que lo haga en grados, de todas maneras, recordar que la equivalencia es 1·rad = 57,29578° ó 2·π·rad=360°.

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  ASENO                   PIC S9V9999.
 01  ACOSENO                 PIC S9V9999.
 01  ATANG                   PIC S9V9999.
*
 PROCEDURE DIVISION.
*
     MOVE 0 TO ASENO ACOSENO ATANG.
*
     COMPUTE ASENO   = FUNCTION ASIN (0,1234).
     COMPUTE ACOSENO = FUNCTION ACOS (0,1234).
     COMPUTE ATANG   = FUNCTION ATAN (0,1234).
*
```

En éste programa, la variable ASENO valdrá 0,1237, la varibale ACOSENO valdrá 1,4471 y ATANG 0,1228.

FUNCIÓN FACTORIAL

Devuelve el FACTORIAL, (n!), del valor entero numérico, pasado como argumento.

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  FAC                     PIC 9(12) COMP-5.
*
 PROCEDURE DIVISION.
*
     COMPUTE FAC = FUNCTION FACTORIAL (10).
```

En éste programa, la variable FAC, tomará el valor 3.628.800.

FUNCIÓN INTEGER/INTEGER-PART

Devuelve el ENTERO más cercano al valor pasado como argumento. Aunque están consideradas funciones distintas, yo no les he visto nada de eso, en ambos casos, el valor ha sido el mismo. ¿?¿?¿?

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  ENT                      PIC S9(5)V9(4) COMP-5.
*
 PROCEDURE DIVISION.
*
     COMPUTE ENT = FUNCTION INTEGER (12345,6789).
```

En éste programa, la variable ENT, tomará el valor 12345. Tener en cuenta que, si el argumento pasado es positivo, a PWC le dará lo mismo lo que haya en los decimales, los trunca y deja la parte entera sin modificar, es decir 7,89234 se queda en 7 y 7,000001, también se queda en 7, pero, si el argumento pasado es negativo, truncará los decimales y le restará 1 a la parte entera, es decir -7,89234 se queda en 8 y -7,000001, también se queda en 8.

FUNCIÓN INTEGER-OF-DAY

Devuelve el número de días que han pasado desde el 1 de enero de 1601, hasta el año y día del año que pasamos como argumento, por ejemplo, si como argumento pasamos el 2° día del año 1602, (con el argumento 1602002), nos devolverá 367, que corresponde a los 365 días del año 1601 + los 2 días del año 1602, no le he encontrado utilidad a ésta función, pero … ahí está.

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  ENT                      PIC 9(6)COMP-5.
*
 PROCEDURE DIVISION.
*
     COMPUTE ENT = FUNCTION INTEGER-OF-DAY (1999365).
```

En éste programa la variable ENT, tomará el valor 145.731, que son los días que han pasado entre el 1er. día del año 1601 y el último día del año 1999. (el valor 1999365, que pasamos como argumento).

FUNCIÓN LOG/LOG10

Devuelve el LOGARITMO NEPERIANO (LOG) o el LOGARITMO DECIMAL (LOG10) del número pasado como argumento.

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  L                        PIC 9V9999 COMP-5.
 01  L10                      PIC 9V9999 COMP-5.
*
 PROCEDURE DIVISION.
*
     COMPUTE L   = FUNCTION LOG   (100).
     COMPUTE L10 = FUNCTION LOG10 (100).
```

El logaritmo neperiano de 100 es 4,6052, ($\mathbf{e}^{4,6052} = 100$), y el logaritmo decimal de 100 es 2, ($10^2 = 100$).

FUNCIÓN MOD

Devuelve el MÓDULO del segundo valor pasado como argumento del primer valor.(El resto de dividir argumento-1/argumento-2), argumento-1 y argumento-2 han de ser valores enteros.

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  A                        PIC 999 COMP-5 VALUE 100.
```

```
 01  A1                      PIC 999 COMP-5 VALUE 23.
 01  M                       PIC 999V9999 COMP-5.
*
 PROCEDURE DIVISION.
*
     COMPUTE M   = FUNCTION MOD (A A1).
```

En éste programa, la variable M, tomará el valor de 8.

FUNCIÓN NUMVAL/NUMVAL-C

Convierte un número dado como un string en su valor numérico, por ejemplo, “-27,892” lo convierte en -27,892, la diferencia entre NUMVAL y NUMVALC, es que el segundo soporta números con el símbolo monetario.

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  A                       PIC S9(5)V999.
*
 PROCEDURE DIVISION.
*
     COMPUTE A = FUNCTION NUMVAL “-123,45”.
```

En éste programa, la variable A, tomará el valor de -123,450.

FUNCIÓN ORD

Devuelve el número que corresponde a un carácter pasado como argumento. Es la inversa a la función CHAR, explicada anteriormente. Hay que tener en cuenta que, por defecto toma el alfabeto por defecto del ordenador a no ser que hayamos definido alguno en especial en la cláusula `ALPHABET` en la sección `SPECIAL-NAMES` de nuestro programa.

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  ORDEN                   PIC 999.
*
 PROCEDURE DIVISION.
*
     COMPUTE ORDEN = FUNCTION ORD (“A”).
```

En este caso, la variable ORDEN, tomará el valor 66 y, al igual que en la función CHAR, no sé por qué toma el valor ASCII + 1, (el alor ASCII de la letra “A” es 65, no 66).

FUNCIÓN ORD-MAX/ORD-MIN

Devuelve el número de orden del valor máximo (ORD-MAX) o del valor mínimo (ORD-MIN), de los valores pasados como argumento.

```
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
*
 01  A                       PIC 99   BINARY.
 01  A1                      PIC 99   BINARY.
 01  T.
     03                      PIC X(3) VALUE "ABC".
     03                      PIC X(3) VALUE "BCA".
     03                      PIC X(3) VALUE "ACB".
     03                      PIC X(3) VALUE "CBA".
     03                      PIC X(3) VALUE "BAC".
     03                      PIC X(3) VALUE "CAB".
 01  T1 REDEFINES T.
     03  T2                  PIC X(3) OCCURS 6.
*
 PROCEDURE DIVISION.
```

```
*
     COMPUTE A  = FUNCTION ORD-MAX (T2(ALL)).
     COMPUTE A1 = FUNCTION ORD-MIN ("H", "HI", "FOR", "COBOL", "AS", "ZIP").
```

En este caso, la variable A tomará el valor 4 y la variable A1, el valor 5.

FUNCIÓN RANDOM

Devuelve un número pseudoaleatorio entre 0 y 1, es pseudoaleatorio porque si no modificamos lo que se conoce como "la semilla", (el argumento que pasamos a la función), siempre nos devolverá el mismo valor, aunque cerremos el programa y lo volvamos a abrir. Para generar un número entre 1 y un valor máximo, la fórmula sería:

random(semilla) x valor_máximo + 1

la fórmula *random(semilla) x valor_máximo*, devuelve un número entre 0 y (valor_máximo – 1), por eso, para completar hasta el máximo, al final le sumamos 1 al resultado.

```
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
*
 01  M                       PIC S9999V99999999999 COMP-5.
 01  Z                       PIC -ZZZZZZ9,99999999999.
 01  HORA                    PIC 9(8).
*
 PROCEDURE DIVISION.
*
     PERFORM WITH NO LIMIT
             ACCEPT HORA FROM TIME
             COMPUTE M = FUNCTION INTEGER ((FUNCTION RANDOM (HORA) * 1000) + 1)
             IF M = 500
                EXIT PERFORM
             ELSE
                MOVE M TO Z
                DISPLAY Z
             END-IF
     END-PERFORM.
*
```

Éste programa, muestra en pantalla un número al azar, entre **1** y **1000**, hasta que coincide con **500** y, en ese momento, se termina el bucle. Para que la "semilla", sea lo más aleatoria posible, cargamos en ésta la hora con milisegundos incluidos, pero para que veáis que con una misma "semilla", se produce un mismos valor aleatorio, debido a la velocidad de cálculo de los ordenadores de hoy en día, cuando aceptamos la hora, aunque sea con milisegundos como es en este caso, y ejecutamos la función, en un milisegundo, se ejecuta varias veces la instrucción random, por lo que tenemos varias veces el mismo valor.

Si lo que queremos es que el número aleatorio, esté entre un máximo y un mínimo, también lo podemos hacer con unas pequeñas y simples modificaciones al programa anterior.

```
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
*
 01  MAX                     PIC S999 VALUE 750.
 01  MIN                     PIC S999 VALUE 250.
 01  M                       PIC S9999V99999999999 COMP-5.
 01  Z                       PIC -ZZZZZZ9,99999999999.
 01  HORA                    PIC 9(8).
*
 PROCEDURE DIVISION.
*
     PERFORM WITH NO LIMIT
             ACCEPT HORA FROM TIME
             COMPUTE M = FUNCTION INTEGER (((FUNCTION RANDOM (HORA) * (MIN - MAX)) +
1) + MAX)
             IF M = 500
                EXIT PERFORM
             ELSE
                MOVE M TO Z
```

```
                DISPLAY Z
              END-IF
      END-PERFORM.
```

Tener en cuenta que, la función RANDOM, es una función lineal en una recta, por lo que el valor aleatorio que nos dé, irá en incremento, hasta que supere “el limite” de la recta, momento en el que “empezará” otra vez por el principio de la recta. Si os fijáis en los valores que os da el programa, éstos van en aumento desde el valor mínimo hasta el máximo, y si no fuera por la condición de fin de bucle en 500, veríais que al llegar al máximo, empieza de nuevo por el mínimo.

FUNCIÓN RANGE

Devuelve la diferencia, (resta), entre el valor máximo y el valor mínimo, de una serie de valores pasados como argumentos de la función.

```
 DATA             DIVISION.
 WORKING-STORAGE  SECTION.
*
 01   M                        PIC S9999V99999999999 COMP-5.
 01   Z                        PIC -ZZZZZZ9,99999999999.
*
 PROCEDURE DIVISION.
*
      COMPUTE M = FUNCTION RANGE (60 30 50 70 90 80 40 45 75).
*
```

Con éste programa, la variable M, tomará el valor 60, (90 - 30)

FUNCIÓN SQRT

Devuelve la raíz cuadrada del número pasado como argumento de la función.

```
 DATA             DIVISION.
 WORKING-STORAGE  SECTION.
*
 01   M                        PIC S9999V99999999999 COMP-5.
*
 PROCEDURE DIVISION.
*
      COMPUTE M = FUNCTION SQRT (12345).
*
```

En éste caso, la variable M, tomará el valor 111,10805551354. Es el equivalente a elevar el argumento a ½ para la raíz cuadrada o a ⅓ para la raíz cúbica, etc. (En COBOL sería `COMPUTE M = numero_positivo ** (1/2).` *o* `COMPUTE M = numero_positivo ** (1/3).` Recordar que en COBOL, para elevar un número a otro, no se utiliza el acento circunflejo (**^**), si no, la doble multiplicación seguida (******) y que en matemáticas, podemos calcular cualquier raíz (cuadrada, cúbica, cuarta …), elevando el número a 1/raíz.

FUNCIÓN CHAR

Devuelve el carácter ASCII correspondiente al valor numérico pasado como argumento. Equivaldría a CHR$() de BASIC.

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  CAR                PIC X VALUE SPACES.
*
 PROCEDURE DIVISION.
*
     MOVE FUNCTION CHAR (66) TO CAR.
```

En éste programa, la variable CAR, tendrá el valor **A**, no sé por qué, pero el valor del argumento, ha de ser el del carácter ASCII + 1, si os fijáis en el ejemplo, para darle el valor **A** a la variable CAR, hemos de pasar como argumento el número 66, cuando todos sabemos que la letra A, su valor ASCII es 65.

FUNCIÓN LOWER-CASE/UPPER-CASE

Convierte un texto pasado como argumento en minúsculas o mayúsculas.

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  ORIGINAL                PIC X(10) VALUE "Hola Mundo".
 01  CONVERTIDO              PIC X(10) VALUE SPACES.
*
 PROCEDURE DIVISION.
*
     MOVE FUNCTION LOWER-CASE (ORIGINAL) TO CONVERTIDO.
     MOVE FUNCTION UPPER-CASE (ORIGINAL) TO CONVERTIDO.
```

En éste programa, en el primer caso la variable CONVERTIDO, toma el valor “hola mundo” y en el segundo caso, toma el valor “HOLA MUNDO”.

FUNCIÓN REVERSE

Invierte un texto pasado como argumento de la función.

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  ORIGINAL                PIC X(10) VALUE "Hola Mundo".
 01  INVERTIDO               PIC X(10) VALUE SPACES.
*
 PROCEDURE DIVISION.
*
     MOVE FUNCTION REVERSE (ORIGINAL) TO INVERTIDO.
```

En éste programa, la variable INVERTIDO, tomará el valor “odnuM aloH”.

FUNCIÓN WHEN-COMPILED

Devuelve la fecha y hora de compilación del programa.

```
 DATA             DIVISION.
 WORKING-STORAGE  SECTION.
*
 01  COMPILADO.
     03  FECHA.
         05  AA              PIC 9(4).
         05  MM              PIC 99.
         05  DD              PIC 99.
```

```
     03  HORA.
         05  HH               PIC 99.
         05  MI               PIC 99.
         05  SS               PIC 99.
*
 01  COMPILADO-EDI.
     03  FECHA-EDI.
         05                   PIC X(22) VALUE "Fecha de compilación:".
         05  DD               PIC 99.
         05                   PIC X VALUE "/".
         05  MM               PIC 99.
         05                   PIC X VALUE "/".
         05  AA               PIC 9999.
     03  HORA-EDI.
         05                   PIC X(9) VALUE " - Hora:".
         05  HH               PIC 99.
         05                   PIC X VALUE ":".
         05  MI               PIC 99.
         05                   PIC X VALUE "'".
         05  SS               PIC 99.
*
 PROCEDURE DIVISION.
*
     MOVE FUNCTION WHEN-COMPILED TO COMPILADO.
     MOVE CORR FECHA TO FECHA-EDI.
     MOVE CORR HORA  TO HORA-EDI.
*
```

En éste programa, en la variable COMPILADO-EDI, tendremos la fecha y hora de compilación de nuestro programa. La variable que recibe los datos, (COMPILADO en éste caso), aunque aquí sólo tenga en total una PICTURE DE 14 "posiciones", puede tener hasta 21 "posiciones" con los siguientes valores:

Posición	Valor
1 a 4	Año en 4 dígitos
5 a 6	Mes (de 01 a 12)
7 a 8	Día (de 01 a 31)
9 a 10	Hora (de 00 a 23)
11 a 12	Minutos (de 00 a 59)
13 a 14	Segundos (de 00 a 59)
15 a 16	Centésimas de segundo [(1)] (de 00 a 99)
17	Signo + ó - [(2)]
18 a 19	Horas de diferencia con Greenwich
20 a 21	Minutos de diferencia con Greenwich.

[(1)] Será siempre 0, (cero), si el S.O. no puede "leer" las centésimas de segundo.
[(2)] De la posición 17 a 21, será siempre 00000, si el sistema no puede "leer" la diferencia horaria con Greenwich. El signo será negativo (**-**) si el país se encuentra a la izquierda (al Oeste) del meridiano de Greenwich (Londres) y tendrá un valor de 0 a 12, y será positivo (**+**) si se encuentra a la derecha (al Este) del meridiano, con un valor posible entre 0 y 13.

Hasta aquí la mayoría de las funciones, creo que hay algunas más, pero, o no las recuerdo ahora o no sé su uso exacto, así que, si alguien me quiere refrescar la memoria, ya sabe, un correo y listo. También deciros que no todas las funciones están disponibles para todas las versiones, contra mayor sea la versión que tengáis del compilador, mayor será el número de funciones disponibles.

Se acabó y Contacto …

Bueno, hasta aquí éste manual, os habréis fijado en que lo mío es programar y no escribir libros, así que pido disculpas por la mala redacción, pero es lo que hay.

Ésta es la versión 1.1.3 del manual, tendrá sus errores como todo, errores que no he cometido a propósito, así que si descubrís alguno/algunos, os rogaría que me los remitieseis a la dirección de correo **josber@gmail.com**, poniendo en el asunto del correo "***error encontrado en manual***", e intentaré corregirlo lo antes posible. Estos errores pueden comprender tanto en la explicación de un apartado, como en la programación del código fuente, que, aunque está probado, (o testeado, para ser más moderno), no implica que no haya posibles fallos.

Así mismo, os agradezco si me remitís cualquier sugerencia, cambio o añadido al manual, a la misma dirección de correo, pero poniendo en el asunto del correo "***sugerencia para manual***", incluido la descripción de los dos o tres controles que no he usado nunca y no he explicado para no meter la pata. ***Muy Importante:*** respetar el texto del asunto de correo, tal y como lo he escrito en ambos casos, o el selector de spam, lo borrará, cuidado con eso.

Todas las correcciones, modificaciones, sugerencias, etc. serán propiedad de su autor y mía, pero sobre todo de su autor, mía será sólo la transcripción a éste manual, y así será especificado y nombrado en su correspondiente apartado, lo que quiere decir que "***… y al César lo que es del César.***", todo el mundo que aporte algo interesante, tendrá su reconocido crédito en el manual. Si queréis que aparezca vuestro correo electrónico junto a vuestro nombre, deberéis indicarlo clara y explícitamente en el correo que me enviéis.

Créditos:

Fujitsu PowerCobol ® es una marca registrada de GT Software inc. ®
Windows, XP, 7, 8, 8.1 y 10 son marcas registradas de Microsoft Corp. ®
Solaris ® es una marca registrada de Oracle Corp. ©
Microsoft Excel, Word y SQL, son marcas registradas de Microsoft Corp. ®
MariaDb es una marca registrada de MariaDB ®
HeidiSQL es propiedad de Ansgar Becker.
Todas las capturas de pantalla, se han realizado con lightshot 5.4.0.1, propiedad de Skillbrains © , por cierto, uno de los mejores y más sencillos programas de captura de pantalla que he manejado.
LibreOffice es una marca registrada de The Document Foundation.

Diario de cambios (o changelog para los modernos)

***** Marzo 2018**:

	*** Lanzada la versión definitiva **1.0.1**	→ **JosBer**
*****Abril 2018:**		
	*** Se añade un índice en las primeras páginas	→ **JosBer**
*****Abril 2018**		
	*** Versión **1.1.0**	→ **JosBer**
	*** Añadidas las rutinas CBL_ más usadas	→ **JosBer** (Gracias BegoGuay por la sugerencia)
	*** Añadidas las Funciones (FUNCTION) más usadas	→ **JosBer** (Gracias BegoGuay por la sugerencia)
*****Mayo 2018:**		
	***Versión **1.1.3**	→ **JosBer**
	*** Añadidas casi todas las Funciones (FUNCTION)	→ **JosBer**
	***Corregidos errores menores con el índice	→ **JosBer**