

# Curso de Power Cobol



Nuestros programas Cobol, tienen todos un aspecto muy similar, las divisiones, los párrafos y sobre todo un desarrollo secuencial de las acciones que debe de realizar. Ahora todo eso cambia un poco y el peso fuerte de la programación lo llevan los llamados eventos, que no son otra cosa que los procesos a realizar cuando se actúa cualquiera de los componentes u objetos que tengamos definidos en nuestra pantalla. Vamos a empezar por dar una pequeña explicación de todo lo que nos vamos a encontrar en adelante y con lo cual deberemos de empezar a familiarizarnos.

- **Proyecto:** Es la base para empezar a desarrollar nuestras aplicaciones y en ellos estarán todas las pantallas, las imágenes e iconos.
- **Objeto:** Será todo aquello que definamos gráficamente en nuestro programa, serán objetos un Push-Button, un Label, un Combobox, una Tabla, una Imagen, etc .... Cada objeto deberá de tener un nombre como si de una variable se tratara. En principio el programa asignará un nombre por defecto pero nosotros podremos variarlo y darle el que mas nos apetezca, ese nombre será el que utilicemos para todo lo que haga referencia a ese objeto. Las pantallas que diseñemos serán los contenedores de objetos, es decir donde definamos todos los objetos que deseemos en nuestra aplicación. Pero a su vez también será un objeto con sus propiedades, sus métodos y sus eventos. En cada aplicación puede haber mas de una pantalla, pero una será siempre la principal.
- **Propiedades:** Serán las distintas opciones que puede tener un objeto y pueden ser comunes o distintas según el tipo de objeto. Serán propiedades, el color, la altura, la anchura, el título, el tipo de letra, si está o no disponible, si está o no visible, etc ... Las propiedades suelen tener un nombre pre-definido por el lenguaje que lo haya designado. Muchas de las propiedades tendrán un valor de tipo SI-NO. Estas propiedades normalmente se podrán establecer tanto en tiempo de diseño como en tiempo de ejecución, es decir, cuando se diseña la pantalla y desde el control del programa, esto es una gran ventaja como veremos mas adelante.
- **Métodos:** Son procedimientos que ya vienen programados por el lenguaje y nosotros solo tendremos que llamarlos para que actúen. Son métodos, añadir campos a un Combobox, enviar el foco a cualquier objeto, abrir una ventana, cerrarla, etc ... Como veremos cada uno de los objetos puede tener sus propios métodos.
- **Eventos:** Serán las acciones del usuario sobre el programa. Como el click sobre un Push-Button, pulsar Return o Tab en un Edit, etc .... Al producirse el evento, se ejecutará todo lo que le hayamos programado y actuará en consecuencia.

Todo esto, lo que nos va a suponer, es que vamos a perder un poco el "control" sobre nuestro programa, puesto que ya no lo vamos a ver como siempre en un editor y de una manera secuencial, sino que cada evento y propiedades tendrán que ser vistas por separado. Pero os aseguro que eso no es un inconveniente, digamos que al principio es un poco chocante. En los próximos capítulos iré explicando el funcionamiento de PowerCobol en su versión 3, la compilación, la ejecución, los objetos o controles, las propiedades y todo lo necesario para generar nuestras aplicaciones en éste entorno de desarrollo.



Al abrir el Fujitsu PowerCobol (en adelante Power), nos sale automáticamente una ventana denominada SHEET en blanco y las ventanas de control que tengamos por defecto, pueden ser la de Status, la de Font, la de Color y la de Item, Si alguna no se abre o bien otra queremos cerrarla, bastará con activarla en el menu Tool.

Sobre ésta primera ventana ya podemos empezar a poner nuestros controles u objetos y darles vida. Pero antes de nada aprendamos mas acerca de la ventana, que en realidad es otro objeto mas de nuestra aplicación.

Veamos cuales son sus propiedades.

- **Sheet name:** Será el nombre que la identificará en nuestra aplicación.
- **Title:** Será lo que aparezca en ella cuando ejecutemos el programa.
- **Icon Name y Cursor Name:** Tendremos ocasión de escoger un tipo de icono para mostrar en la barra de título, puede ser uno de los que trae por defecto o cualquiera que nosotros hayamos creado o vinculado. Y el cursor igualmente.
- **Window, Frame, Style:** Con ellas podremos variar la apariencia visual de nuestra ventana. Lo mas recomendable es que probéis y os quedéis con la que mas os guste.

Con la ventana de Status, obtenemos información acerca de la posición relativa de nuestra ventana con respecto a la pantalla y el tamaño de anchura y altura que tiene. Así mismo, con la ventana de color podemos modificar el color de fondo y de texto que tendrá.

En cuanto a los eventos que tiene un objeto de tipo "sheet" o ventana, son los siguientes:

- **SPECIAL-NAME:** Actuará igual que en un programa normal, es decir aquí definiremos lo mismo que haríamos en un programa normal en nuestra CONFIGURATION SECTION. También es posible definir tipos de impresión y otras posibilidades que nos ofrece el propio compilador de Fujitsu.
- **FILE-CONTROL:** Definición de archivos que vamos a utilizar en nuestra ventana Exactamente igual que se explican en los manuales con la excepción de que no hay que especificar el tipo de dispositivo excepto en la definición de la impresora que si se pondrá. Además aqui es donde si queremos especificar que vamos a trabajar con un fichero Btrieve (RM) se le indica con las letras BTRV, después del ASSIGN TO. Un ejemplo sería el siguiente:

```
SELECT SOCIOS ASSIGN TO "SOCIOS.DAT" BTRV  
ORGANIZATION INDEXES ACCESS DYNAMIC  
RECORD KEY KEYSOC  
ALTERNATE RECORD KEY KEYSOC1  
FILE STATUS STASOC.
```

```
SELECT IMPRE ASSIGN TO PRINTER.
```

## Curso de Power Cobol

- **BASED:** Es algo propio del Fujitsu y no se para que sirve, además en lo que he podido leer no he conseguido nada, puede ser que sea por compatibilidad con sus productos anteriores, de todas formas no lo he necesitado para realizar ninguna aplicación.
- **FILE:** Aquí será donde se definan las descripciones de los archivos que vayamos a utilizar. Quiero hacer incapié en una cosa. Tened siempre en cuenta que cuando trabajemos con Power, las variables serán por defecto locales y por lo tanto no se extenderán al resto de ventanas que utilice nuestra aplicación, para ello es necesario utilizar la opción GLOBAL y EXTERNAL. De tal modo que una FD quedaría:

```
FD SOCIOS GLOBAL EXTERNAL LABEL RECORD STANDARD.  
01 REGSOC.  
02 KEYSOC.  
03 .....
```

- **WORKING:** Que os voy a contar, aquí definiremos las variables a utilizar en el programa. Recordad lo que os he dicho antes, podéis utilizar GLOBAL y EXTERNAL si lo creéis conveniente, para usarlas en otras ventanas de la aplicación. Además quiero añadir que por ejemplo, si luego algún componente determinado va a utilizar alguna variable exclusiva la podremos definir para ese componente y no para el resto de programa. Recordad que la ventana sobre la que estamos hablando es otro componente de nuestro programa para POWER. Eso significa también que si definimos una variable como: 01 SALUDO PIC X(30). Esa variable no la podremos utilizar en un campo de nuestra ventana, ya que al no definirla como GLOBAL, le indicamos al compilador que solo se utilizará en la ventana como componente, pero no en el resto del programa, por lo cual lo mas razonable es definir las todas como GLOBAL. El ponerle EXTERNAL, hará la misma función que conseguimos con la LINKAGE SECTION.
- **CONSTANT:** Para definir constantes, pero se pueden definir igualmente en la Working como siempre hemos hecho con VALUE.
- **PROCEDURE:** iii OJO !!! Atención, aquí se van a definir las rutinas que luego podremos llamar desde nuestro programa, en ningún caso, lo que aquí se ponga, se ejecutará por si solo. Yo me tiré dos días preguntándome porque no se ejecutaban las sentencias que ponía aquí. Imaginaros las rutinas que hacéis para luego llamarlas con CALL, pues bien eso mismo es esto,, hasta hay que definir las completamente es decir con IDENTIFICATION, PROGRAM-ID, etc ..  
Un ejemplo:

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. LIMPIAR IS COMMON.  
ENVIRONMENT DIVISION.  
DATA DIVISION.  
PROCEDURE DIVISION.  
INICIO SECTION.  
MOVE 0 TO POW-NUMERIC OF CODIGO.  
MOVE SPACES TO POW-TEXT OF NOMBRE.  
MOVE SPACES TO POW-TEXT OF DIRECCION.  
MOVE SPACES TO POW-TEXT OF POBLACION.  
EXIT PROGRAM.
```

# Curso de Power Cobol

## END PROGRAM LIMPIAR.

Si veis la estructura es similar a un pequeño programa. Para luego llamarlo desde cualquier parte, simplemente CALL "LIMPIAR" en este caso porque ese es su nombre. En este caso lo que hará será mover a espacios y a ceros el contenido de los controles que se indican (CODIGO, NOMBRE, DIRECCION y POBLACION).

- **OPENED:** Aquí si pondremos realmente lo que queramos que el programa realice justo al mostrar la ventana, por ejemplo, rellenar los combobox o grid que tengamos, o abrir fichero o cualquier acción previa a la visualización de la ventana. Podremos tener aquí nuestra propia Environment o Data Division si fuese necesario.
- **CLOSE:** Aquí se darán las ordenes que el programa ejecute, justo al cerrar la ventana, lo mas normal será cerrar los ficheros, pero podremos dar cualquier comando.
- **CLOSECHILD:** En este apartado, programaremos las acciones que el compilador ejecutará siempre que cerremos una ventana que haya sido abierta desde ésta. Es decir si desde ésta ventana principal llamamos a otra. Justo al cerrarse esta nueva y antes de pasar el control a la principal se ejecutará este apartado.

Una vez explicado esto podemos empezar a colocar los objetos que deseemos en nuestra ventana. Recordad todo bien, en la PROCEDURE irán las rutinas que queramos llamar luego desde el programa, en OPENED, irá lo que deseemos que el programa realice antes de mostrar la ventana. Y en toda la DATA, cualquier dato que queramos que sea portable a cualquiera de los componentes del programa, lo declararemos con GLOBAL y si además queremos que sea portable a otras ventanas, además de GLOBAL, le pondremos EXTERNAL. Esto es algo muy importante.

Tenéis que tener en cuenta que cada ventana es un programa independiente (por llamarlo de alguna forma) es decir que los datos y variables si no las definís con EXTERNAL, no se corresponderán. En Acucobol o RM, siempre que abrimos una ventana desde un programa, éste sigue teniendo el control y no hay porque definir nada nuevo, aquí NO, aquí cada ventana es un programa.

Solo los mensajes que podremos displayar en ventanas pequeñas, del tipo (SI / NO), (ACEPTAR / CANCELAR ) formarán parte de nuestro programa o ventana.

Una vez creada la ventana, lo siguiente que debemos de hacer es incorporarla a nuestro Proyecto y estará lista para ser ejecutada.

En el siguiente curso, voy a explicar como se llaman las variables que hacen referencia a las propiedades de los controles mas usuales, para después realizar un pequeño programa y ver como funciona.

# Curso de Power Cobol

**Objeto:** Será todo aquello que definamos gráficamente en nuestro programa, serán objetos un Push-Button, un Label, un Combobox, una Tabla, una Imagen, etc .... Cada objeto deberá de tener un nombre como si de una variable se tratara. En principio el programa asignará un nombre por defecto pero nosotros podremos variarlo y darle el que mas nos apetezca, ese nombre será el que utilizemos para todo lo que haga referencia a ese objeto. Las pantallas que diseñemos serán los contenedores de objetos, es decir donde definamos todos los objetos que deseemos en nuestra aplicación. Pero a su vez también será un objeto con sus propiedades, sus métodos y sus eventos. En cada aplicación puede haber mas de una pantalla, pero una será siempre la principal.

Veamos una lista de los objetos que nos presenta PowerCobol, en ésta versión 3 y una pequeña explicación de para que sirven.

Al final de la página explicaré para que sirven las diferentes opciones de estilo que tienen los objetos, para casi todos son las mismas. Y cuando actúa cada evento

OBJETO	DESCRIPCION	EVENTOS
	<b>LABEL:</b> Un Label, es una etiqueta, un comentario o cualquier cosa que queramos poner en nuestro programa en texto. Sustituye a los DISPLAY (entre comillados) de nuestros programas.	- CLICK - CHANGE
	<b>EDIT:</b> Un Edit nos sirve para aceptar cualquier tipo de campo. Sustituye a los ACCEPT de nuestro programa.	- CHANGE - EDIT - RETURN
	<b>PICTURE EDIT:</b> Un Picture Edit, es igual a un Edit, pero con la ventaja de poder aceptar el campo con la edición que le definamos mediante un PIC. Como desventaja, perdemos el evento CHANGE.	- RETURN - EDIT
	<b>PUSH BUTTON:</b> Un Push Button, es un botón que se puede pinchar para ejecutar cualquier cosa. No se podría decir exactamente a que sustituye en nuestros programas, pero casi siempre a las preguntas de aceptación.	- CLICK
	<b>CHECK BUTTON:</b> Un Check Button, nos sirve para escoger o no una opción. Puede sustituir a las típicas preguntas de Si o No, sobre algunas opciones de nuestros programas.	- CLICK
	<b>RADIO BUTTON:</b> Un Radio Button suele ir acompañado de otros y nos	- CLICK

## Curso de Power Cobol

	<p>sirve para escoger entre varias opciones, una posible. Siempre la usaremos cuando las opciones estén claramente delimitadas.</p>	
	<p><b>GROUP BOX:</b> Un Group Box, es un rectángulo que puede contener un título. Nos sirve para agrupar controles que mas o menos tienen algo que ver entre si. Por ejemplo un grupo de Radui Button.</p>	
	<p><b>LIST BOX:</b> Un List Box, es una lista desplegable con opciones de entre las cuales podemos escoger una. Por ejemplo, para escoger una de las provincias del país.</p>	<ul style="list-style-type: none"> <li>- DBLCLICK</li> <li>- SELCHANGE</li> </ul>
	<p><b>COMBO BOX:</b> Un Combo Box, es una mezcla de Edit y List Box. Aunque se puede presentar de varias formas. Yo la he utilizado mas que el List Box.</p>	<ul style="list-style-type: none"> <li>- CHANGE</li> <li>- SELCHANGE</li> <li>- EDIT</li> <li>- RETURN</li> </ul>
	<p><b>HORIZONTAL SCROLL BAR:</b> Una Barra de Scroll horizontal. La podemos utilizar para presentar el progreso de algo que estemos realizando, siempre que sepamos los valores de inicio y fin.</p>	<ul style="list-style-type: none"> <li>- CHANGE</li> <li>- ENDSCROLL</li> </ul>
	<p><b>VERTICAL SCROLL BAR:</b> Igual que la anterior, solo que la presentación es vertical, en vez de horizontal.</p>	<ul style="list-style-type: none"> <li>- CHANGE</li> <li>- ENDSCROLL</li> </ul>
	<p><b>TIMER:</b> Un control para asignar espacios de tiempo programables.</p>	<ul style="list-style-type: none"> <li>- TIMER</li> </ul>
	<p><b>TABLE:</b> Magnifico control con el cual creamos una tabla dimensionable. Las barras de desplazamiento se ponen automaticamente dependiendo del tamaño que le hayamos dado.</p>	<ul style="list-style-type: none"> <li>- CLICK</li> <li>- DBLCLICK</li> <li>- EDIT</li> <li>- RETURN</li> </ul>
	<p><b>GRAPH:</b> Nos permite presentar un gráfico de barras sobre unos valores dados previamente. En este caso es de una dimension unicamente.</p>	
	<p><b>DDE:</b> Dyanmic Data Exchange, para comunicaciones con otras aplicaciones.</p>	<ul style="list-style-type: none"> <li>- DDECHANGE</li> </ul>
	<p><b>SOUND:</b> Permite incluir un fichero de sonido con formato WAV.</p>	
	<p><b>FUNCTION KEY:</b> Igual que un Push Button, pero a la que se le puede asignar una tecla de función.</p>	<ul style="list-style-type: none"> <li>- CLICK</li> </ul>
	<p><b>DATE:</b> Un control que nos sirve para poner la fecha actual en diferentes</p>	<ul style="list-style-type: none"> <li>- CLICK</li> <li>- CHANGE</li> </ul>

## Curso de Power Cobol

	formatos.	
	<b>METAFILE:</b> Podemos colocar archivos de tipo .emf, .wmf, .clp. No he podido comprobar su uso.	- CLICK - CHANGE
	<b>SIMPLE ANIMATION:</b> Podemos hacer una animacion, indicando varios archivos graficos tipo BMP e indicando el intervalo entre cada uno.	- CLICK - DBCLICK - STARTANIME - ENDANIME
	<b>MCI:</b> Control utilizado para exponer videos AVI, o poder hacer un reproductor de CD-Audio.	
	<b>RECTANGLE:</b> Para dibujar un rectangulo en nuestras pantallas. Puede tener su sentido para dar vistosidad o crearnos barras de herramientas.	
	<b>DRIVE LIST:</b> Control para mostrarnos y utilizar las unidades que tengamos en nuestro disco.	- SELCHANGE
	<b>DIRECTORY LIST:</b> Para mostrarnos los directorios o carpetas de una determinada unidad.	- CHANGE - SELCHANGE
	<b>FILE LIST:</b> Para mostrar los archivos de un directorio seleccionado, podemos escoger entre sus atributos.	- DBCLICK - SELCHANGE
	<b>BITMAP BUTTON:</b> Un Push Button, al cual podemos incorporar una imagen para mayor vistosidad.	- CLICK
	<b>SELECTION BOX:</b> Es igual que un ComboBox, en el que las opciones se cargan directamente desde un archivo.	- SELCHANGE - EDIT - RETURN
	<b>EXCEL CONNECTION:</b> Permite enlazar y obtener datos de una hoja de calculo en Excel. En esta versión falla, porque la familia Office, a la que pertenece Excel se instala en el directorio Archivos de Programa, y ya sabeis los problemas de esta version con los directorios que contienen espacios en blanco.	
	<b>PRINT:</b> Control con el que podemos imprimir los campos de la pantalla actual que tengan activada la casilla de printable. Muy bueno. Digamos que es como un Imprimir Pantalla bastante mejorado.	
	<b>EXTEND IMAGE:</b> Igual que el Image, pero que admite mas versiones de ficheros de imagenes.	- CLICK - DBCLICK - CHANGE

## Curso de Power Cobol



**DB ACCESS:** Control que nos permite enlazar con una base de datos y manejar las tablas en nuestro programa de Cobol.

Ya sabéis que pulsando sobre el botón derecho del ratón y siempre que estemos sobre un objeto, nos aparecerá un menú con el cual nos podremos dirigir tanto al estilo del objeto como a las procedures de cada evento.

Los campos que tenemos dentro de Style, son casi iguales en todos los casos, voy a explicar los mas comunes:

- **Item Name:** Indica el nombre con el que el programa se referirá a dicho componente, ese nombre aunque sale por defecto, podemos modificarlo a nuestro antojo.
- **Text:** Hace referencia al texto que saldrá por defecto puesto en el componente.
- **Visible:** Indica si el objeto será visible en el momento de la ejecución.
- **Enable:** Indica si el objeto estará disponible en el momento de la ejecución.
- **3-D:** Indica si el objeto tendrá una presentación en 3-D, es decir con bordes resaltados y creando una sensación de efecto de 3 dimensiones.
- **Border:** Indica si el campo tendrá un borde rodeándole cuando aparezca.
- **Tab Stop:** Indica si cuando pulsemos Tab, el cursor se posicionará en alguna ocasión en este objeto o por el contrario nunca lo hará.
- **Print:** Indica si este objeto será impreso cuando se seleccione la orden de imprimir con el objeto Print.
- Además encontraremos en la mayoría la alineación tanto vertical como horizontal del componente.

Por supuesto cada uno de estos estilos son modificables en tiempo de ejecución, lo que le definamos aquí será para la primera vez que aparezcan, porque de alguna manera tienen que aparecer, pero nosotros podemos modificar tantas veces como queramos. En el siguiente capítulo explicaré como se hace referencia a los estilos y como se actúa con ellos.

Los eventos, serán las acciones que se produzcan cuando se actúe sobre alguno de ellos, por ejemplo el evento de un Push-Button, será el Click, es decir, cuando hagamos click con el ratón sobre el. Al programar un evento, lo que le decimos al programa es lo que tiene que hacer cuando se produzca dicho evento.

- **CLICK:** Este evento se produce al pinchar con el botón izquierdo del ratón sobre un objeto.
- **DBCLICK:** Este evento se produce al hacer doble-click con el botón izquierdo del ratón sobre un objeto.
- **CHANGE:** Este evento se produce cuando cambia el contenido de

# Curso de Power Cobol

- un objeto.
- **SELCHANGE:** Este evento se produce cuando cambia la selección de un objeto de tipo lista, es decir cuando cambiamos en un ListBox, o en un Combobox.
- **EDIT:** Se produce cuando se entra en edición en un campo de recogida de datos, Edit, Picture-Edit, Table, etc...
- **RETURN:** Se produce cuando pulsamos Return sobre un objeto. Hay que tener en cuenta que en muchos de los objetos podemos definir que otra acción puede hacer que se produzca el evento RETURN, por ejemplo al perder el foco.

Respecto a versiones posteriores (yo ahora tengo la 5), hay que hacer notar que los eventos aumentan en número considerable y que los objetos son mas y algunos se han integrado con otros.

---

Esta sección se actualizó por última vez el 16 de Noviembre de 2.000.

[Ir al principio de la página](#) 



**Propiedad:** Cada una de las opciones que puede tener un objeto y pueden ser comunes o distintas según el tipo de objeto. Serán propiedades, el color, la altura, la anchura, el título, el tipo de letra, si está o no disponible, si está o no visible, etc ... Las propiedades suelen tener un nombre pre-definido por el lenguaje que lo haya designado. Muchas de las propiedades tendrán un valor de tipo SI-NO.

Las propiedades se ajustan normalmente en tiempo de diseño, pero por muchos motivos será necesario cambiarlas también en tiempo de ejecución, para cambiar estas propiedades utilizaremos nuestro comando de COBOL, MOVE. De tal manera que lo haremos igual que cuando le damos un valor a una variable.

El formato para referirnos a las propiedades de un objeto será el siguiente:

**MOVE** valor **TO** propiedad **OF** control.

En el caso de PowerCobol, las nombres de las propiedades están predefinidas y todas empiezan por la palabra POW- seguida del nombre de la propiedad. En las propiedades de tipo SI-NO, podemos utilizar para el "NO" el valor "0" o también la palabra reservada de PowerCobol, POW-OFF, para el caso del "SI" podremos utilizar "1" o también POW-ON.

A continuación voy a explicar y poner algunos ejemplos de las mas utilizadas.

PROPIEDAD	DESCRIPCION	VALORES
POW-ACTIVATE	Solo se aplica al objeto timer y sirve para iniciarlo o pararlo. <b>MOVE POW-ON TO POW-ACTIVATE OF RELOJ.</b>	- POW-ON - POW-OFF
POW-BACKCOLOR	Para indicar el color de fondo de un objeto, aplicable a la mayoría de ellos. Podemos utilizar POW-RED (rojo), POW-BLUE (azul), etc.. También para colorear celdas de una tabla. <b>MOVE POW-RED TO POW-BACKCOLOR OF NOMBRE.</b> <b>MOVE POW-BLUE TO POW-BACKCOLOR (1,1) OF TABLA.</b>	- POW-RED - POW-BLUE - POW-BLACK - etc .....
POW-BORDER	Para que aparezca el objeto con borde, Label, Edit, Image, y algún otro. <b>MOVE POW-ON TO POW-BORDERE OF CAMPO.</b>	- POW-ON - POW-OFF
POW-CHECK	En los objetos RadioButton, CheckButton y Menu, indica si esta o	- POW-ON

## Curso de Power Cobol

	no chequeado. <b>MOVE POW-ON TO POW-CHECK OF ACTIVO.</b>	- POW-OFF
POW-COLS	Contiene el número de columnas de una tabla. <b>MOVE POW-COLS OF TABLA TO COLUMNAS.</b>	- Valor numérico. (devuelve valor)
POW-COUNT	En los ListBox, ComboBox y demás objetos con listas, nos devuelve el número de elementos que tiene. <b>MOVE POW-COUNT OF LISTA1 TO ELEMENTOS.</b>	- Valor numérico. (devuelve valor)
POW-DATA	En los Graph, el valor de cada una de los elementos del gráfico. <b>MOVE 30 TO POW-DATA (3) OF GRAFICO.</b>	- Valor numérico.
POW-DATACOLOR	En los Graph, el color de cada una de los elementos del gráfico. <b>MOVE POW-RED TO POW-DATACOLOR (3) OF GRAFICO.</b>	- POW-RED (resto de colores)
POW-ENABLE	En todos los objetos, para ponerlos en activos o pasivos. <b>MOVE POW-ON TO POW-ENABLE OF GRAFICO.</b>	- POW-ON - POW-OFF
POW-NUMERIC	Introduce un valor numérico en los objetos PictureEdit y Table. <b>MOVE 14 TO POW-NUMERIC(3,3) OF TABLA.</b>	- Valor numérico.
POW-PRNENABLE	En todos los objetos, indican si van a ser imprimibles con el objeto Print. <b>MOVE 1 TO POW-PRNENABLE OF NOMBRE.</b>	- POW-ON - POW-OFF
POW-ROW	Nos devuelve el número de línea que tiene el foco en una tabla. <b>MOVE POW-ROW OF TABLA TO LINEA.</b>	- Valor numérico. (devuelve valor)
POW-ROWS	Le indicamos el número de líneas que tiene una tabla. <b>MOVE 130 TO POW-ROWS OF TABLA.</b>	- Valor numérico.
POW-SELECT	En los objetos de listas ListBox, ComboBox, nos indica el elemento que está seleccionado. <b>MOVE POW-SELECT OF LISTA TO ELEMENTO. MOVE 1 TO POW-SELECT OF LISTA.</b>	- Valor numérico.
POW-TEXT	En muchos controles, nos indica el título o el contenido del mismo. <b>MOVE "TITULO" OF POW-TEXT OF ETIQUETA.</b>	- POW-RED (resto de colores)
POW-TEXTCOLOR	En muchos controles, nos indica el color del texto. <b>MOVE POW-GREEN TO POW-TEXTCOLOR OF ETIQUETA.</b>	- Valor alfanumérico.
POW-VISIBLE	En casi todos los objetos, nos sirve para indicar si están o no visibles en nuestras ventanas. <b>MOVE 1 TO POW-VISIBLE OF GRAFICO.</b>	- POW-ON - POW-OFF

Existen muchas mas propiedades, pero se irán viendo en el ejemplo que empezaremos a desarrollar. Este capítulo está mas enfocado hacia el modo de introducción de datos en las propiedades que a la explicación de todas ellas.

Con estas propiedades además de hacer el MOVE, se puede programar, es decir, por ejemplo podemos preguntar por ellas:

**IF POW-SELECT OF LISTA = 1 MOVE "HA SELECCIONADO LA OPCION 1" TO POW-TEXT OF ETIQUETA.  
IF POW-SELECT OF LISTA = 2 MOVE "HA SELECCIONADO LA OPCION " TO POW-TEXT OF ETIQUETA.**

Del mismo modo que también podemos coger los datos de esas propiedades para en nuestro programa hacer con ellos lo que deseemos. Por ejemplo:

**MOVE POW-TEXT OF ETIQUETA TO NOMBRE.**

Con estos ejemplos simples, tendríamos que al final la variable NOMBRE definida en nuestra WORKING, como alfanumérica tendría el valor dependiendo de la opción seleccionada en el ListBox llamado LISTA. Y también el objeto Label llamado Etiqueta contendría ese mismo valor.

El próximo capítulo estará dedicado a la utilización de los métodos y luego empezaré con el desarrollo de un programa completo, igual que se hizo en la sección programando, es decir, tendremos la misma agenda, pero ahora en formato gráfico de

Windows.

Esta sección se actualizó por última vez el 22 de Enero de 2.001.

[Ir al principio de la página](#) 



**Método:** Son procedimientos que ya vienen programados por el lenguaje y nosotros solo tendremos que llamarlos para que actúen. Son métodos, añadir campos a un Combobox, enviar el foco a cualquier objeto, abrir una ventana, cerrarla, etc ... Como veremos cada uno de los objetos puede tener sus propios métodos.

Para acceder a todos los métodos se utilizará la sentencia CALL, quedando su formato de la siguiente manera:

**CALL método OF objeto USING parámetros.**

A continuación vamos a explicar algunos de los métodos mas habituales. Pero como siempre os digo, en los ejemplos prácticos es donde mejor se comprende todo.

METODO	DESCRIPCION
ADDSTRING ADDSTRING256	Para añadir elementos a objetos ListBox y ComboBox. <b>CALL ADDSTRING OF LISTA USING "PRIMER ELEMENTO".</b>
ALARM	Aplicable solo en los Sheet y produce un sonido, podemos escoger entre varios predefinidos.POW-MBOK, POW-MBASTERISK, POW-MBQUESTION, POW-MBEXCLAMATION, POW-MBHAND. <b>CALL ALARM OF SHEET1 USING POW-MBOK.</b>
CLEARLIST	Aplicable a ListBox y ComboBox, conseguimos reiniciar los objetos de tal manera que quedan vacios completamente. <b>CALL CLEARLIST OF LISTA.</b>
OPENPRINTER WHITESHEET CLOSEPRINTER	Solo para el objeto Print. Abrimos y cerramos el objeto Print, para provocar una salida por impresora de los objetos de nuestra pantalla que tengan la opción PrnEnable activada. <b>CALL OPENPRINTER OF IMPRESORA.</b> <b>CALL WRITESHEET OF VENTANA1.</b> <b>CALL CLOSEPRINTER OF IMPRESORA.</b>
CLOSESHEET	Aplicable solo a las ventanas, con ello las cerramos. <b>CALL CLOSESHEET OF VENTANA1.</b>
DELETESTRING	Aplicable a los controles ListBox y ComboBox, y conseguimos borrar un elemento de la lista. <b>CALL DELETESTRING OF LISTA USING 1.</b>
DISPLAYMESSAGE	Se va a explicar en la parte final. Por ser un método muy utilizado y util.
GETCELLNUMERIC GETCELLTEXT	Aplicable a las tablas y conseguimos extraer el contenido de las celdas, ya sean numéricas o alfanuméricas. En el ejemplo moveremos a la variable VALOR, el contenido de la celda situada en la columna 1 línea 1 de la tabla llamada TABLA. <b>CALL GETCELLTEXT OF TABLA USING VALOR 1 1.</b>
OPENSHEET	Aplicable solo a las Sheet y con ella las abrimos. Las ventanas siempre dependen de una padre. <b>CALL OPENSHEET OF PRINCIPAL USING "SEGUNDA".</b>
SETCELLNUMERIC SETCELLTEXT	Aplicable a las tablas y conseguimos introducir un valor en las celdas, ya sean numéricas o alfanuméricas. En el ejemplo moveremos el valor "HOLA" a la celda situada en la columna 1 línea 1 de la tabla llamada TABLA. <b>CALL SETCELLTEXT OF TABLA USING "HOLA" 1 1.</b>
SETFOCUS	Aplicable a la mayoría de los controles. Con el conseguimos pasar el foco de un objeto a otro. <b>CALL SETFOCUS OF LISTA.</b>

Existen muchos mas métodos, pero estos son los mas utilizados. Veamos como un

## Curso de Power Cobol

ejemplo sencillo como rellenaríamos un ListBox a partir de una tabla definida en la Working. Siendo LISTA el nombre que tiene nuestro ComboBox o ListBox. Y además tenemos una tabla con dos columnas, en la primera introducimos el número y en la segunda su nombre.

```
WORKING-STORAGE SECTION.
01 TABLA.
    02 FILLER PIC X(10) VALUE "PRIMERO".
    02 FILLER PIC X(10) VALUE "SEGUNDO".
    02 FILLER PIC X(10) VALUE "TERCERO".
    02 FILLER PIC X(10) VALUE "CUARTO".
    02 FILLER PIC X(10) VALUE "QUINTO".
01 TABLITA REDEFINES TABLA.
    02 ELEMEN PIC X(10) OCCURS 5 TIMES.
01 CONTA PIC 9.
01 CAMPO PIC X(10).
...
PROCEDURE DIVISION.
INICIO.
    MOVE 0 TO CONTA.
UNO.
    ADD 1 TO CONTA IF CONTA > 5 GO DOS.
* aqui cargamos el ComboBox.
    CALL ADDSTRING OF LISTA USING ELEMEN (CONTA).
* aqui cargamos la tabla.
    CALL SETCELLNUMERIC OF TABLA USING CONTA CONTA 1.
    MOVE ELEMEN (CONTA) TO CAMPO.
* en la version 3 de PowerCobol da error si introducimos un campo
* con subindice directamente en una tabla, por eso primero lo
* muevo a un campo.
    CALL SETCELLTEXT OF TABLA USING CAMPO CONTA 2.
    GO UNO.
DOS.
    EXIT.
```

Después de ejecutar el programa:

el ComboBox nos hubiera quedado así:



Y la tabla hubiera quedado así:

1	PRIMERO
2	SEGUNDO
3	TERCERO
4	CUARTO
5	QUINTO

### **DISPLAYMESSAGE**

Con este método de PowerCobol, conseguimos que se nos muestre una ventana independiente pero sin perder el control sobre la que tenemos activa. Son las típicas ventanas de confirmación de Windows y tienen unos parámetros pre-asignados. A continuación vamos a ver algunos ejemplos y como quedan en la práctica. Su formato, como cualquier otro método es el siguiente:

```
CALL DISPLAYMESSAGE OF nombredeventana USING texto título estilo.
```

## Curso de Power Cobol

- **Texto:** el texto hace referencia al texto que nos saldrá al mostrar el mensaje en la ventana.
- **Título:** el título hace referencia al título que tendrá la venta que nos saldrá.
- **Estilo:** El estilo se refiere al icono que tendrá la ventana y a las posibles opciones que se nos presenten, en cuanto a los ICONOS:
  - POW-DMNOICON, sin icono.
  - POW-DMICONSTOP, icono con el símbolo Stop.
  - POW-DMICONQUESTION, icono de interrogación.
  - POW-DMICONEXCLAMATION, icono con el signo de exclamación.
  - POW-DMICONINFORMATION, icono con una (i).
- **Estilo:** En cuanto a los botones:
  - POW-DMOK, solo el botón de Ok.
  - POW-DMOKCANCEL, botones de Ok y Cancelar.
  - POW-DMABORTRETRYIGNORE, botones de Abortar, Reintentar e Ignorar.
  - POW-DMYESNOCANCEL, botones de Si, No y Cancelar.
  - POW-DMYESNO, botones de Si y No.
  - POW-DMRETRYCANCEL, botones de Reintentar y Cancelar.
- **Respuestas:** las posibles respuestas al pulsar los distintos botones.
  - POW-DMROK, si hemos pulsado el boton de Ok.
  - POW-DMRCANCEL, si pulsamos Cancelar.
  - POW-DMRABORT, si pulsamos Abortar.
  - POW-DMRRETRY, si pulsamos Reintentar.
  - POW-DMRIGNORE, si pulsamos ignorar.
  - POW-DMRYES, si pulsamos Si.
  - POW-DMRNO, si pulsamos No.

El estilo se debe de guardar en una variable con el formato: 01 ESTILO PIC S9(4) COMP-5. Y para introducir los valores que deseemos lo haremos de la siguiente manera:

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

De esta manera asignamos al estilo el icono de Interrogación y los botones de Si y No, quedando nuestra orden completa de la siguiente manera, teniendo la ventana donde saldrá el nombre de VENTANA1:

```
WORKING-STORAGE SECTION.
```

```
01 ESTILO PIC S9(4) COMP-5.
```

```
01 TITULO PIC X(20).
```

```
01 TEXTO PIC X(40).
```

```
...
```

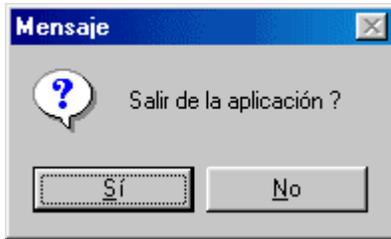
```
PROCEDURE DIVISION.
```

```
INICIO.
```

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

```
MOVE "Mensaje" TO TITULO.
```

## Curso de Power Cobol



MOVE "Salir de la aplicación ?" TO  
TEXTO.

CALL DISPLAYMESSAGE OF  
VENTANA1 USING TITULO TEXTO  
ESTILO.

....

Ahora nos saldrá la correspondiente  
ventana con el mensaje y entonces puede que le demos al  
botón del Si o al del No, para controlarlo utilizamos la  
siguiente instrucción:

```
IF PROGRAM-STATUS = POW-DMRYES  
  MOVE "HAS PULSADO SI" TO POW-TEXT  
  OF TEXTODEEXPLICACION.  
IF PROGRAM-STATUS = POW-DMRNO  
  MOVE "HAS PULSADO NO" TO POW-TEXT  
  OF TEXTODEEXPLICACION.
```

---

Esta sección se actualizó por última vez el 22  
de Enero de 2.001.



# Curso de Power Cobol



## **INTRODUCCION.**

Ya tenemos creado nuestro directorio llamado Agenda, dentro de FSC. Esta versión de Power, no nos permite darle con un número el tamaño de la ventana, sino que lo conseguimos con el ratón agrandando o disminuyendo como cualquier ventana de Windows. Os digo esto porque para conseguir el tamaño de las ventanas que os habéis bajado en el ejecutable, tendréis que hacerlo como podáis.

Las palabras que os encontréis en rojo, harán referencia a puntos del menú de Power.

Lo primero que hacemos es crearnos un proyecto: Project-New lo guardáis en c:\fsc\agenda\agenda.prj.

Si no os habéis bajado todos los iconos que vamos a utilizar en la aplicación, este es un buen momento para hacerlo. Podéis encontrar el enlace al final de la pantalla, debéis descomprimir el archivo totico.zip en la misma carpeta.

Ahora creamos nuestra primera ventana: File-New, una vez abierta le dáis al botón derecho del ratón sobre ella, pinchamos sobre Style y ponemos lo siguiente:

- **Sheet** name: PRIMERA
- **Title**: Agenda v1.0
- **Icon name y Cursor name**: Por ahora no se tocan
- **Window**: Popup Window
- **Frame**: Thin
- **Style**: Title, Minimize Box y Control Menú
- **3D**: sin marcar

Pulsamos Ok y como color de fondo seleccionamos el que mas nos guste, si queréis respetar el que tiene, lo agregáis a la paleta de colores es: 198,198,255.

Una vez realizado hacemos: File-Save as, y le damos el nombre de agenda.win. A continuación nos vamos a Project-Edit, pinchamos sobre Add y añadimos la pantalla al proyecto, como va a ser la pantalla con la que arranque la aplicación marcamos la casilla, Starting Sheet.

## **PRIMERA COMPILACION.**

Ahora ya, podríamos por ejemplo, compilar y ejecutar, aunque obviamente no nos iba a hacer nada, pero Power ya reconoce un proyecto completo. Podéis hacer una prueba, le dáis Project - Build y luego Project - Run.

A partir de ahora para ejecutar bastará solo con dar a Run, si ha habido alguna modificación se compilará automáticamente.

Tenéis que tener en cuenta que Project-Compile, actuará sobre la ventana activa, mientras que Project-Build, compilará todas las ventanas del proyecto.

Project-Link y Project-Make, se encargan de generar el ejecutable para su ejecución. Para un proyecto pequeño podemos hacer siempre Build, pero si es muy largo, para depurar errores iremos compilando ventana a ventana.

Quizás haya sido un poco lioso lo que he explicado o quizás algo demasiado simple que ya todos conocéis, pero creo que siempre es bueno refrescar las ideas y dar la oportunidad de ayudar al que empieza desde la nada.

## **LAS IMAGENES.**

## Curso de Power Cobol

Cuando utilizamos imagenes en PowerCobol, podemos usarlas como imagenes independientes, o bien como un recurso propio de la aplicación, me explico: Una imagen puede ir con la aplicación en un archivo mas y llamarla así en nuestra aplicación o bien que las imagenes se integren en el ejecutable de la aplicación y no vayan como archivos independientes. La diferencia estará en el tamaño del ejecutable, pero en cambio, nadie podrá modificar las imagenes que incluyamos. En la aplicación utilizaremos ambas para que veáis como se implementan.

Vamos a colocar el icono para la aplicación siguiendo los siguientes pasos:

- Project-Edit, Add , en tipo de archivo seleccionamos Icon (\*.ICO) y nos aparecerá el que he creado para ello: Agenda.ico.
- En la casilla Resource name, le ponemos AGENDA. Que será el nombre con el que Power reconocerá dicho icono.
- A continuación nos vamos al Style de nuestra ventana principal y en la casilla Icon Name, ponemos AGENDA. Cuando compilemos y ejecutemos de nuevo ya tendrá nuestra aplicación dicho icono y si miráis con el Explorador de Windows en la carpeta veremos que el archivo AGENDA.EXE, también se identifica con él.

Ahora vamos a colocar la primera imagen en nuestra ventana, para ellos seleccionamos el icono Extend Image de la ventana de controles y lo arrastramos hasta nuestra ventana.

Una vez colocado el icono de la imagen en la ventana, nos vamos a Style y en Image File, ponemos bloc.bmp. Podríamos utilizar también el botón Image File para localizarlo, pero eso nos daría el path completo de donde se encuentra, con lo que si luego cambiamos la aplicación de carpeta, la imagen no saldría, por eso es mejor, siempre que esté en la misma carpeta del proyecto, poner solo su nombre. Además deshabilitaremos la casilla de Border, para que se quede perfectamente enclavado en nuestra pantalla, ahora solo nos queda ir probando para asignar el tamaño preciso, tanto a la ventana como a la imagen, en cualquier momento podremos pulsar el botón de Test y comprobar como va quedando.

### **CREANDO EL MENU.**

Abrimos la ventana de menubar y en title colocamos lo que se verá en la barra de menú: &Opciones.

Pulsamos Enter y ahora escribimos &Consulta y le damos al botón que tiene los signos => para implementarlo dentro de Opciones.

Pulsamos sobre el botón de Extend, para que nos muestre mas opciones para el menú y pinchamos donde indica Separator: Ins Under, que significa que vamos a introducir un separador del menú debajo de la opción de Consultas.

Ahora pinchamos sobre Ins Under en la ventana de la izquierda para introducir otra opción en el menú a la que llamamos &Acerca de ...

A continuación volvemos a introducir un separador igual que lo hemos hecho antes y para concluir volvemos a pulsar sobre Ins Under en la ventana de la izquierda para introducir otra opción, en esta caso la última a la que vamos a llamar &Salir.

A continuación cerramos la ventana de MenuBar y ya podemos darle al test para comprobar como nos ha quedado el menu.



Esta sección se actualizó por última vez el 23 de Abril de 2.001.



## **CONTROLES**

Este es el momento de empezar a colocar los objetos en nuestra pantalla. Esta misión de diseño no es la mas importante, pero si va a ser la mas atractiva para nuestro usuario o cliente, que va a querer que la información se presente de manera clara y ordenada, por eso nada mejor que utilizar la utilidad del Grid, para que los controles se alineen con mayor facilidad en la pantalla.

Los controles que vamos a utilizar en esta pantalla son:

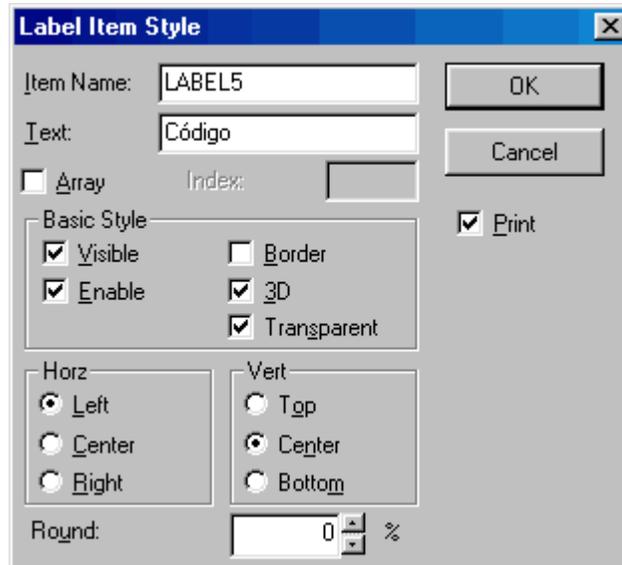
- **Label:** Serán lo que siempre poníamos con DISPLAY. Con ellas pondremos todos los nombres de los campos que vamos a introducir, Código, Nombre, Domicilio, Población, etc .... Podéis ponerlo donde queráis, sin dejaros ninguno o ponerlo como yo lo he hecho en el ejemplo.
- **Group Box:** Son las cajas que facilitan la agrupación de controles, aunque en esta versión solo sirven para eso, para decorar, clarifica aún más la información.
- **Picture Edit:** Son las cajas donde vamos a aceptar los datos, es decir lo que sustituye a los ACCEPT. Sin duda lo mas importante y el mas utilizado.
- **Radio Button:** Botones de opción para elegir entre varios, por eso he añadido el campo género.
- **Combo Box:** Caja de opciones, lo he utilizado para señalar el tipo de contacto.
- **Bitmap Button:** Son los botones que harán los procesos.

Una vez los tengáis colocados, es el momento de asignarles un nombre, ese nombre es importante y hará referencia al control. Los nombres si debemos de darlos igual, para que luego en la programación no haya problemas.

### **LABEL**

A los Label con el nombre del campo se le puede dejar el nombre que se le asigne PowerCobol. Mejor veamos esta imagen que corresponde al Label de Código.

## Curso de Power Cobol

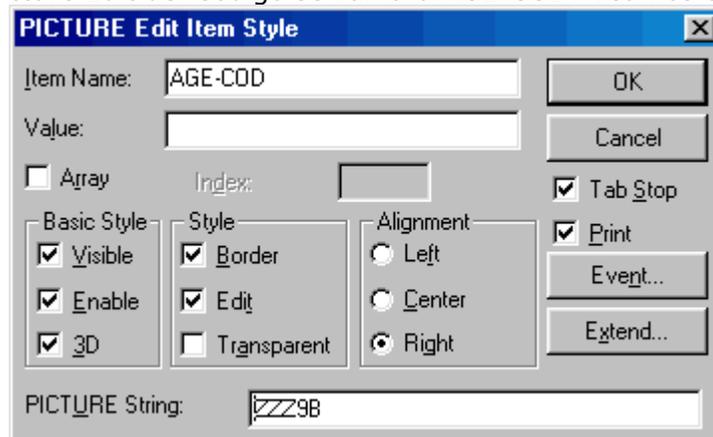


Así es como deben de ir todos los Labels, solo cambiando el Text, que hará referencia a cada uno de nuestros campos.

### **PICTURE EDIT**

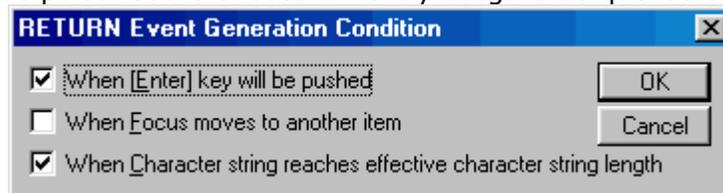
En los Picture Edit, donde introduciremos los campos, su nombre será el mismo que el nombre del campo del registro pero con un guión enmedio.

Por ejemplo el Picture Edit del Código se llamará AGE-COD. Veamos otra imagen.



Aquí fijaros que algo muy importante es PICTURE String, que nos indicará como se mostrará el dato una vez introducido. Por eso he elegido éste control y no el Edit normal. En este caso PIC ZZZ9B.

Además debéis de pulsar sobre el botón Event y aseguraros que las casillas estén así:



Con ésta ventana le indicamos cuando se efectuará el evento Return, es decir cuando el campo se dará por aceptado. La primera casilla indica que lo hará al pulsar ENTER (activada). La segunda que lo hará cuando el foco esté en otro control (desactivada). La tercera indica que se producirá cuando se rellene todo el campo con su longitud (activada).

## Curso de Power Cobol

Con esto conseguimos que los Picture Edit, funcionen igual que los ACCEPT tradicionales.

A continuación pongo los valores de todos los estilos de los Picture Edit:

Campo	Nombre	Picture String
Código	AGE-COD	ZZZ9B
Nombre	AGE-NOM	X(30)
Domicilio	AGE-DOM	X(30)
Población	AGE-POB	X(20)
Código Postal	AGE-POS	9(5)B
Provincia	AGE-PRO	X(15)
Teléfono	AGE-TEL	X(20)
Teléfono Móvil	AGE-MOV	X(20)
e-mail	AGE-MAI	X(30)
Página Web	AGE-WEB	X(40)

### **OTROS**

Para el Género creamos dos Radio Button, al que identifica al hombre lo llamamos AGE-GEH y el que identifica mujer lo llamamos AGE-GEM. Uno de los dos debe de llevar activa la casilla de Check en su estilo, que nos indica cual estará seleccionado por defecto.

Para el Tipo creamos un ComboBox y lo llamamos AGE-TIP. Tendremos en cuenta de marcar la casilla Dropdown List, dentro del estilo del Combo Box.

Los Group Box, como os dije antes son aclaratorios y solo tienen diseño, así que podéis ponerlos como en el ejemplo agrupando campos mas o menos comunes.

### **ICONOS**

Para simular el hueco donde van los iconos, he dibujado un rectángulo (Rectangle) al que le he dado el color de fondo azul marino. Y luego encima he colocado todos los iconos y label de los iconos.

Vamos a incluir en nuestro proyecto todos los iconos que os habéis bajado y que forman parte de la aplicación. Van a ser recursos y por lo tanto van a formar parte de nuestro proyecto, recordad lo que os comenté en el capítulo anterior sobre como tratar las imágenes.

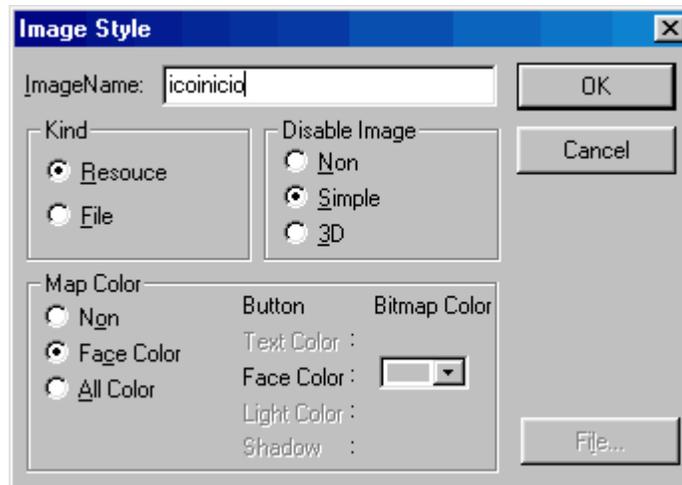
Por lo tanto abrimos nuestro menú Project, nos vamos a Edit y pulsamos sobre Add. Una vez allí en tipo de archivos marcamos Bitmap (\*.bmp) y vamos añadiendo uno a uno. Ya sabéis que cada vez que se añade uno debéis de darle un nombre de recurso. Veamos los que hay que añadir.

Imagen	Nombre recurso
borrar.bmp	icoborrar
consul.bmp	icoconsul
fin.bmp	icofin
grabar.bmp	icograbar
inicio.bmp	icoinicio.bmp
listar.bmp	icolistar
mail	icomail
mas.bmp	icomas
menos.bmp	icomenos
ok.bmp	icook
salir.bmp	icosalir

Ahora es momento para ir colocando los Bitmap Image e ir asignando sus nombres.

## Curso de Power Cobol

Primero creamos los cuatro referentes a la búsqueda de registros y los llamamos: INICIO, MENOS, MAS y FIN, asignándoles los recursos apropiados de icoinicio, icomenos, icomas e icofin. Para asignar el nombre del recurso hay que pinchar sobre Image en la caja de estilo de cada uno de los Bitmap Button: Lo haremos igual con todos.



A continuación colocamos el icono de borrar al que llamamos BORRAR. Luego Ok, al que llamamos OK. Después listar, al que llamamos LISTAR. Seguido de e-mail al que llamamos EMAIL. Luego Grabar (GRABAR), consulta (CONSULTA) y salir (SALIR). Ahora solo nos quedan los nombres de los Botones que se colocan también con label. Los cuatro primeros, los de las flechas de búsqueda, los podemos poner juntos y con el nombre que se le asigne. Pero para los demás lo deberán llevar y será el siguiente, cada uno irá debajo del botón:

texto de Label	Nombre label
Borrar	L-BORRAR
Ok	L-CANCELAR
Listar	L-LISTAR
e-mail	L-EMAIL
Grabar	L-GRABAR
Consulta	L-CONSULTA
Salir	L-SALIR

### **PRINT**

Ahora colocamos el control Print en un lugar de la pantalla que no nos estorbe, este tipo de controles no se ven en tiempo de ejecución, pero es necesario que estén integrados en nuestra ventana, para poder asignarle los valores. Una vez colocado, por ejemplo en la esquina superior derecha nos vamos a su cuadro de estilo y ponemos como nombre PRINT1 y como Title, FICHA DE LA AGENDA. Lo demás lo podéis dejar por defecto.

### **DDE**

Vamos a colocar también un DDE, que es un control que nos sirve para conectar con otras aplicaciones externas y que utilizaremos para enviar e-mail. Igual que el anterior no se ve en tiempo de ejecución, pero tenemos que darle propiedades, que serán muy simples. Tenéis que abrir su caja de estilo y simplemente poner en ItemName, DDE1.

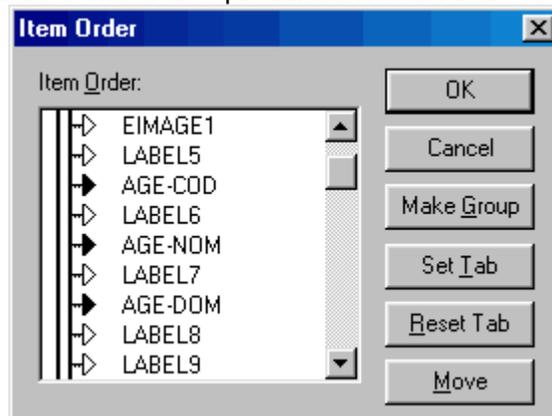
### **ITEM ORDER**

Llegados a este punto solo quedar explicar una pequeña cosa. Todos los componentes que hemos ido poniendo llevan un orden y en algunos casos pudiera suceder que unos no se vean, simplemente porque se han quedado debajo, bien, eso se soluciona,

## Curso de Power Cobol

sacando el menu pop-up sobre cualquier punto de la pantalla y dirigiendonos a la opción que pone Item Order. Ahí debéis de llevar todo lo que se debe de quedar debajo, es decir los Group Box y el Rectangle a una posición mas alta que lo que va a contener.

También debéis de tener en cuenta en marcar los tabuladores sobre los campos de nuestro fichero y no sobre lo demás, además en el mismo orden en que los hemos colocado. Una muestra de como debe quedar se muestra en ésta imagen.



Los que tienen el triángulo en negro, son los controles sobre los que nos moveremos al pulsar la tecla TAB. Se ponen con Set Tab y se quitan con Reset Tab. Dejad solo los que hacen referencia a los Picture Edit.

### **CONCLUSION**

Llegados aquí os hago una pregunta: ¿Vamos bien ?. Es que no es fácil explicar en la distancia y sobre papel y no quisiera que os perdierais o se quedara algo sin explicar con claridad. Si una vez leído todo lo aquí expuesto, probáis a compilar y ejecutar y algo no funciona, me lo comunicais y lo modifico al instante.





## **EMPEZANDO A PROGRAMAR**

Una vez hemos aprendido a diseñar y colocar componentes en nuestras ventanas, ha llegado el momento de empezar a introducir el código fuente necesario para que nuestra aplicación sea algo más que diseño.

Os vuelvo a recordar que esta programación orientada a eventos, es algo diferente de la tradicional y por lo tanto, habrá parte de código que pongamos en nuestra ventana y parte que irá dentro de los eventos de los componentes.

Para empezar vamos a programar la ventana principal. Pinchamos sobre el botón derecho del ratón sobre cualquier parte de la ventana teniendo en cuenta que no estemos sobre ningún componente. Se nos desplegará un menú con las distintas opciones y nosotros pinchamos sobre Procedure.

A continuación se nos muestra una lista de todos los eventos disponibles para la ventana. Vamos a programar. El primer evento en programar será FILE-CONTROL, así que nos situamos sobre el y pinchamos en OK. Para aclararnos un poco, voy a hacer una tabla con los eventos y su programación:

<b>Evento</b>	<b>Programación</b>	<b>Ayuda</b>
FILE-CONTROL	SELECT AGENDA ASSIGN TO "AGENDA.DAT" ORGANIZATION INDEXED ACCESS DYNAMIC RECORD KEY KEYAGE FILE STATUS STAAAGE.	Definición de nuestros archivos.
FILE	FD AGENDA GLOBAL EXTERNAL LABEL RECORD STANDARD. 01 REGAGE. 02 KEYAGE. 03 AGECOD PIC 9999. 02 AGENOM PIC X(30). 02 AGEDOM PIC X(30). 02 AGEPOS PIC 99999. 02 AGEPOB PIC X(20). 02 AGEPRO PIC X(15). 02 AGEGEN PIC X. 02 AGETIP PIC 9. 02 AGETEL PIC X(20). 02 AGEMOV PIC X(20). 02 AGEMAI PIC X(30). 02 AGEWEB PIC X(40).	Descripción de los ficheros a utilizar en nuestra aplicación. Al ponerle GLOBAL EXTERNAL conseguimos que los valores de los campos se pasen de una ventana a otra y por todos los controles de la ventana.
WORKING	01 STAAAGE GLOBAL PIC XX. 01 MENSAJE IS GLOBAL. 02 FILLER PIC X(7) VALUE "ERROR: ". 02 NUMSTA PIC 99BB. 02 FILLER PIC X(15) VALUE "POR EL MOTIVO: ". 02 NOMSTA PIC X(15). 01 TABLA GLOBAL. 02 TCOD PIC 9(4) OCCURS 3000 TIMES. 01 CCC GLOBAL PIC 9999. 01 TOPE GLOBAL PIC 9999. 01 AHIVA PIC 9(4) GLOBAL EXTERNAL.	Variable de estado. Mensaje en caso de error al acceder al fichero.  Tabla las con claves de nuestro fichero. Contador. Tope de contador. Variable.
OPENED	ENVIRONMENT DIVISION. DATA DIVISION. WORKING-STORAGE SECTION. 01 MENSAJE. 02 FILLER PIC X(30) VALUE "SE VA CREAR EL	Lo que se ejecutará justo al abrir la ventana. Como véis lleva su propia Working

## Curso de Power Cobol

	<pre> ARCHIVO AGENDA". 01 TITULO.   02 FILLER PIC X(6) VALUE "ERROR". 01 ESTILO PIC S9(4) COMP-5. PROCEDURE DIVISION. DECLARATIVES. INICIO SECTION.   USE AFTER ERROR PROCEDURE ON AGENDA. END DECLARATIVES.   ADD POW-DMICONEXCLAMATION POW-DMOK   GIVING ESTILO.  OPEN INPUT AGENDA IF STAAGE = "35" CALL DISPLAYMESSAGE OF PRIMERA USING MENSAJE   TITULO ESTILO OPEN OUTPUT AGENDA. CLOSE AGENDA.  OPEN I-O AGENDA. CALL ADDSTRING OF AGE-TIP USING "PERSONAL ". CALL ADDSTRING OF AGE-TIP USING "PROFESIONAL". CALL ADDSTRING OF AGE-TIP USING "FAMILIAR ". CALL ADDSTRING OF AGE-TIP USING "EMPRESAS ". CALL ADDSTRING OF AGE-TIP USING "SANITARIO ". CALL "CARGAR".         </pre>	<p>independiente de la otra.</p> <p>Usamos las Declaratives para controlar el status del fichero.</p> <p>Si el status es 35 significa que no existe y nos saca una ventana indicándonoslo. A continuación abrimos el archivo como I-O y cargamos el combobox con los tipos de contacto que vamos a tener. Hacemos una llamada a la rutina CARGAR.</p>
CLOSE	<pre> ENVIRONMENT DIVISION. DATA DIVISION. PROCEDURE DIVISION.   CLOSE AGENDA.         </pre>	<p>Al cerrar la ventana, también cerramos nuestro fichero. Lo último que se hace.</p>
CLOSECHILD	<pre> ENVIRONMENT DIVISION. DATA DIVISION. WORKING-STORAGE SECTION. 01 ESTILO PIC S9(4) COMP-5. PROCEDURE DIVISION.   MOVE POW-ON TO POW-ENABLE OF PRIMERA.   IF AHIVA &gt; 0 GO SUERTE.   CALL SETFOCUS OF AGE-COD.   EXIT PROGRAM. SUERTE.   MOVE AHIVA TO POW-NUMERIC OF AGE-COD.   MOVE AHIVA TO AGE-COD.   READ AGENDA NOT INVALID KEY CALL "EXISTE".   CALL "PONCCC".   EXIT PROGRAM.         </pre>	<p>Esto se realizará cada vez que se cierre una ventana hija de ésta. La ponemos activa. Comprobamos si nos viene con un código y si es así procedemos a mostrarlo y sino se va de nuevo a pedir el código.</p>

Bien, espero que hayais entendido mas o menos todo lo que hemos introducido. Recordad el evento CLOSE se realiza cuando se cierra la aplicación y el CLOSECHILD, cuando se cierra una de las ventanas hija.

Fijaros como controlamos si el estatus del fichero es 35 para hacer un OPEN OUTPUT y por lo tanto crear el fichero siempre que no exista.

Lo mas complicado de entender debe de ser el evento CLOSECHILD. Si recordáis tenemos una ventana que es de consulta en la cual salen todos nuestros contactos en una tabla y pinchando sobre uno de ellos se va y nos lo muestra en la ventana principal, lo podéis ver en el ejecutable directamente. Pues bien, la varibale ahiva, lo que nos da es el número de contacto sobre el que hemos pinchado dos veces en la consulta, por eso al cerrar la ventana hija comprubo si el valor de ahiva es mayor que 0 y entonces leo el registro seleccionado.

El valor de ahiva puede ser 0 si cerramos la ventana de consulta sin pinchar sobre

# Curso de Power Cobol

ningún contacto o bien si hemos abierto la ventana de Acerca de.. ya que el tratamiento será el mismo porque ambas ventanas son hijas de la principal.

Si seguís teniendo alguna duda, antes de continuar me lo decís y lo explico mejor. La idea es que vosotros mismo podáis realizar una aplicación similar a ésta y para ello es mejor que comprendáis todo.

---

Esta sección se actualizó por última vez el 29 de Mayo de 2.001.

[Ir al principio de la página](#)



## **PROGRAMANDO LAS RUTINAS**

Ya sabéis que cuando programamos con PowerCobol, en realidad estamos

cuando algún evento ha entrado en acción, por ejemplo al hacer clic con el ratón sobre algún Push Button, al introducir un texto en un Picture Edit, etc ... Pero también es posible que algunos de esos procesos se vayan a repetir en mas de una ocasión, entonces el hecho de copiarlos en cada evento no nos estaría facilitando la programación.

Para ello utiliza Power la Procedure de cada Sheet para introducir los procesos que no son propiedad de algún evento en concreto. En realidad son como pequeños programas alojados dentro del propio programa, ya que cada uno debe de tener un inicio con IDENTIFICATION DIVISION y un fin con END PROGRAM. Pero todas irán seguidas una debajo de otra dentro de la PROCEDURE DIVISION de la Sheet en que se vayan a utilizar.

En nuestro caso vamos a tener cuatro procedures, a continuación paso a explicaros el porque de cada una y su programación. Quiero hacer desde aquí una reflexión, ya que es un tema que nos puede interesar a todos, yo personalmente programo muy poco en programación estructurada, porque, por mi costumbre, porque nunca he programado en grupo y en la mayoría de los casos he sido autodidacta, por lo tanto me gustaría dejar claro que cada uno puede realizar los procesos como mejor sepa o como mejor se adapte a su manera de programar, yo utilizo el Go y para los ejemplos tengo que hacer uso de el.

### **RUTINA "EXISTE"**

La primera se llama EXISTE y va a ser llamada siempre que introduzcamos un código de contacto que exista en nuestro fichero. Ya habéis que no existe en la aplicación un mantenimiento como tal, es decir con altas, bajas, consultas y modificaciones sino, que simplemente dependiendo del código que introduzcamos haremos una acción u otra. Si al introducir un código, éste no existe, nos permitirá darlo de alta y si por el contrario existe, se nos visualizará en pantalla y podremos modificarlo, consultarlo o borrarlo.

En nuestro caso, además al comenzar todos los campos de nuestro mantenimiento están desactivados para obligar a que solo se pueda introducir el código. Por eso en ésta rutina además de mover el contenido del registro a los controles de la pantalla,

## Curso de Power Cobol

los ponemos en enable para poder entrar en ellos y además el botón de Grabar pasa a tener el texto Regrabar para indicar que se va a modificar.

Esta sería su programación.

```

Programación
IDENTIFICATION DIVISION.
PROGRAM-ID. EXISTE IS COMMON.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
INICIO SECTION.
    MOVE 0 TO POW-ENABLE OF AGE-COD.
    MOVE AGECOD TO POW-NUMERIC OF AGE-COD.
    MOVE AGENOM TO POW-TEXT OF AGE-NOM.
    MOVE AGEDOM TO POW-TEXT OF AGE-DOM.
    MOVE AGEPOS TO POW-NUMERIC OF AGE-POS.
    MOVE AGEPOB TO POW-TEXT OF AGE-POB.
    MOVE AGEPRO TO POW-TEXT OF AGE-PRO.
    IF AGEGEN = "H" MOVE POW-ON TO POW-CHECK OF AGE-GEH
        ELSE MOVE POW-ON TO POW-CHECK OF AGE-GEM.
    MOVE AGETIP TO POW-SELECT OF AGE-TIP.
    MOVE AGETEL TO POW-TEXT OF AGE-TEL.
    MOVE AGEMOV TO POW-TEXT OF AGE-MOV.
    MOVE AGEMAI TO POW-TEXT OF AGE-MAI.
    MOVE AGEWEB TO POW-TEXT OF AGE-WEB.
    MOVE 1 TO POW-ENABLE OF GRABAR.
    MOVE 1 TO POW-ENABLE OF L-GRABAR.
    MOVE 1 TO POW-ENABLE OF BORRAR.
    MOVE 1 TO POW-ENABLE OF L-BORRAR.
    MOVE 1 TO POW-ENABLE OF LISTAR.
    MOVE 1 TO POW-ENABLE OF L-LISTAR.
    MOVE 1 TO POW-ENABLE OF EMAIL.
    MOVE 1 TO POW-ENABLE OF L-EMAIL.
    MOVE 1 TO POW-ENABLE OF CANCELAR.
    MOVE 1 TO POW-ENABLE OF L-CANCELAR.
    MOVE 1 TO POW-ENABLE OF AGE-NOM.
    MOVE 1 TO POW-ENABLE OF AGE-DOM.
    MOVE 1 TO POW-ENABLE OF AGE-POB.
    MOVE 1 TO POW-ENABLE OF AGE-POS.
    MOVE 1 TO POW-ENABLE OF AGE-PRO.
    MOVE 1 TO POW-ENABLE OF AGE-GEH.
    MOVE 1 TO POW-ENABLE OF AGE-GEM.
    MOVE 1 TO POW-ENABLE OF AGE-TIP.
    MOVE 1 TO POW-ENABLE OF AGE-TEL.
    MOVE 1 TO POW-ENABLE OF AGE-MOV.
    MOVE 1 TO POW-ENABLE OF AGE-MAI.
    MOVE 1 TO POW-ENABLE OF AGE-WEB.
    MOVE "Regrabar" TO POW-TEXT OF L-GRABAR.
    CALL SETFOCUS OF CANCELAR.
    EXIT PROGRAM.
END PROGRAM EXISTE.
```

### **RUTINA "CARGAR"**

Esta segunda es una rutina un poco particular y es que a mi, me gusta siempre que voy a trabajar con un fichero y por supuesto siempre que no sea enormemente grande, me gusta cargarlo en una tabla y así el recorrido a través de él lo hago de manera mas rápida y segura. Con ellos conseguimos saber el número de registros que tenemos, al movernos hacia delante o hacia atrás ir con seguridad a los registros y la lectura siempre se hace directa sobre el registro en cuestión puesto que lo tenemos

## Curso de Power Cobol

siempre en la tabla.

Como véis es una simple rutina con un contador y una tabla de un elemento para ir cargando los códigos de nuestra agenda. Al final guardamos en la variable tope el número de registros totales.

Esta rutina será llamada siempre que se haga un cambio en el fichero, es decir cuando demos un nuevo contacto de alta o cuando borremos alguno, para que la tabla siempre sea el reflejo del fichero. Esta acción es sumamente rápida en ningún caso ralentizará el programa.

Esta sería su programación.

```
Programación
IDENTIFICATION DIVISION.
PROGRAM-ID. CARGAR IS COMMON.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
INICIO SECTION.
    MOVE 0 TO CCC.
    CLOSE AGENDA.
    OPEN INPUT AGENDA.
    INITIALIZE TABLA.
UNO.
    READ AGENDA NEXT RECORD AT END GO DOS.
    ADD 1 TO CCC MOVE AGECD TO TCD (CCC).
    GO UNO.
DOS.
    CLOSE AGENDA.
    OPEN I-O AGENDA.
    MOVE CCC TO TOPE MOVE 0 TO CCC.
    MOVE 1 TO POW-ENABLE OF AGE-COD.
    CALL SETFOCUS OF AGE-COD.
    EXIT PROGRAM.
END PROGRAM CARGAR.
```

### **RUTINA "PARASALIR"**

Esta rutina tiene poco que explicar, simplemente se encarga de sacar una ventana en la pantalla en la que nos pregunta si estamos de acuerdo en abandonar la aplicación.

Esta rutina la he puesto porque hay mas de un sitio desde el cual podemos salir, desde un botón en la pantalla principal y desde una opción del menú.

Esta sería su programación.

```
Programación
IDENTIFICATION DIVISION.
PROGRAM-ID. PARASALIR IS COMMON.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
```

# Curso de Power Cobol

```
01 ESTILO PIC S9(4) COMP-5.
PROCEDURE DIVISION.
    ADD POW-DMYESNO POW-DMICONQUESTION GIVING ESTILO
    CALL DISPLAYMESSAGE OF PRIMERA USING "Salir de la aplicación ?" "Mensaje"
ESTILO.
    IF PROGRAM-STATUS = POW-DMRYES GO UNO ELSE GO FINALIZAR.
    UNO.
        CALL CLOSESHEET OF PRIMERA.
        EXIT PROGRAM.
    FINALIZAR.
        EXIT PROGRAM.
END PROGRAM PARASALIR.
```

## RUTINA "PONCCC"

Perdonad por los nombres, es la fuerza de la costumbre. Pero en realidad esta rutina lo que hace es tener siempre localizado el elemento de la tabla que estamos utilizando. Es decir siempre que consultemos cualquier código la tabla también debe de posicionarse en ese punto. De esa manera en el momento en que pulsemos el botón de siguiente o anterior registro, lo haga teniendo como referencia el registro último que hemos consultado.

Así que lo único que hace es recorrer la table en busca del registro actual.

Esta sería su programación.

```
Programación
IDENTIFICATION DIVISION.
PROGRAM-ID. PONCCC IS COMMON.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
INICIO SECTION.
    PERFORM VARYING CCC FROM 1 BY 1
        UNTIL CCC = TOPE OR TCOD (CCC) = AGHECOD
    END-PERFORM.
EXIT PROGRAM.
END PROGRAM PONCCC.
```

Finalizadas las explicaciones sobre la Sheet en los siguientes capítulos veremos la programación de los eventos de los controles que haya en pantalla. Como véis hasta ahora siempre hemos estado utilizando código perfectamente familiar para nosotros, por lo que podemos decir que el hecho de programar en Power no nos hace abandonar nuestro lenguaje habitual.