GUIBUILDER
MANUAL DE USUARIO

Manual preparado en Costa Rica por Daniel Ortíz de SPS Software <docsps@racsa.co.cr>
y Jorge Garro de UniSoft <unisoft@racsa.co.cr>

Manual preparado en Costa Rica por Daniel Ortíz de SPS Software <docsps@racsa.co.cr>
y Jorge Garro de UniSoft <unisoft@racsa.co.cr>

¿ Qué es GUIBuilder ?

GUIBuilder es una herramienta que simplifica enormemente el desarrollo de programas GUI permitiéndole concentrarse en reglas, en lugar de los detalles de cómo escribir un programa GUI. Él automáticamente construye un armazón que incluye el completo loop de eventos, con lugares donde usted puede insertar código, conocido como manejador de eventos, para responder a los eventos seleccionados.
Hay varias ventajas de desarrollar programas usando GUIBuilder:

Centro de Trabajo

Usted puede diseñar, implementar, depurar, y probar los programas GUI sin dejar nunca el centro de trabajo de GUIBuilder.

Visualmente Orientado

El proceso entero de diseñar y llevar a cabo un programa en GUIBuilder es visualmente orientado. Usted dibuja la interface del usuario en ResBuilder, entonces da click en el control seleccionado para asociar código de Visual PRO/5 con eventos específicos.

Generador de Programa

GUIBuilder genera la estructura global para el manejo de eventos de un programa GUI. Usted únicamente tiene que proporcionar el código para manejar cualquier evento en que usted este interesado.

Funciones de Dirección de datos

Cuando usted define los elementos de interface de usuario en ResBuilder, usted asigna nombres a las ventanas y a controles. Los programas de GUIBuilder incluyen una serie de funciones estandar que le permiten que se refiera a controles usando esos nombres en lugar de los números del control.

Diseño GUI

La parte más dificil sobre el programar GUI no es la mecánica de manejo de ventanas y controles; con una herramienta como GUIBuilder, mucho de los detalles mecánicos pueden automatizarse. La dificultad real está en aprender a pensar de una forma completamente nueva. En un programa GUI, usted no controla el flujo del programa directamente; usted crea una ventana con varios controles gráficos y espera por el usuario para hacer algo. Con tal de que un control sea visible y habilitado, el usuario puede activarlo (por ejemplo, empujando un botón, seleccionando un ítem del menú, verificando una caja de chequeo o radio-boton, o entrando texto en un edit-control). Su programa necesita estar preparado para responder de una manera lógica a lo que el usuario escoge hacer. Su control directo sobre el proceso está limitado. Uno de los acercamientos más útiles es hacer sólo ventanas y

controles visibles y activos cuando usted está preparado para responder a ellos. Por ejemplo, si no hay ninguna impresora configurada, usted puede deshabilitar las funciones de impresora relacionadas. El usuario puede seleccionar sólo la opción de la Impresión cuando usted sabe que la impresora está disponible, el manejo de error es mucho más simple.

GUIBuilder hace un buen trabajo administrando los detalles técnicos del manejo de eventos de un programa GUI, pero no puede ayudar con el diseño del programa. Porque un programa GUI opera muy diferente de un programa de modo de carácter tradicional, es una buena idea invertir algún tiempo leyendo y pensando sobre diseño GUI. Las referencias siguientes son un punto de arranque útil:

The Windows Interface Guidelines for Software Design (Microsoft):
http://www.microsoft.com/win32dev/uiguide/

Macintosh Human Interface Guidelines (Apple):
http://developer.apple.com/techpubs/mac/HIGuidelines/HIGuidelines-2.html

Interface Hall of Shame, Interface Hall of Fame, and useful links:
http://www.iarchitect.com/

Articles in The Advantage:
http://www.basis.com/advantage/mag-v1n4/resources.html
http://www.basis.com/advantage/mag-v1n4/offtheshelf.html

GUIBuilder Listas Desplegables

Las listas desplegables proporcionan la capacidad para accesar y revisar bloques de código.

Función de Lista desplegable

Object

Ésta es una lista de objetos de alto-nivel donde el programador escoja, incluya:
- Inicialización (revisar código de inicialización),
- Formas (una selección para cada forma o child window en el recurso),
- Fin del Programa (revisar código de fin de programa)
- Subrutinas (una selección para cada subroutine/function definido en el archivo .gbf), y
- Nuevo Subroutine/Function. Las otras tres listas desplegables son sólo funcionales si el usuario selecciona una Forma ID en esta lista.

Window

Si el usuario seleccionara una Forma en la lista de Objeto, esto contendrá una lista de ventanas definida para esa forma. De primero en la lista esta la propia forma, seguida por

child windows, si no hay ningun child window, la caja de lista de window mostrará el ID de la Forma, y no será editable por el usuario.

Control

Después de que una ventana se ha seleccionado en la lista window, el usuario puede escoger el control en la lista de controles. De primero en la lista de controles sea la ventana (o forma), porque pueden definirse eventos para las ventanas. siguiendo el form/window se verá la lista de controles en esa ventana. Si la ventana no contiene ningún control, los form/window se seleccionarán automáticamente, y no será editable por el usuario.

Event

Después de que un control se ha seleccionado en la lista de controles, el usuario puede escoger un evento en la lista de Evento. Si hay sólo un evento definido para el control seleccionado (ej. un tool button), el evento se selecciona automáticamente, y el usuario va directamente a la ventana de edición para crear o editar el código del evento. Si hay más de un posible evento para el control seleccionado, usted puede seleccionar uno de la lista.
Un " * " (asterisco) delante del evento significa que una rutina de evento ya existe para ese código de evento. Si la lista de evento dice"-----ningún Evento Definido-----", entonces el control seleccionado no tiene ningún evento (ej. una caja de grupo, "group box"). Si la lista de evento dice "-----ningún Evento Visible-----", entonces el control seleccionado define por lo menos un posible evento, pero los eventos definidos para este control no estan visibles debido a las propiedades de evento puestas en el archivo de recurso. presione el boton de la barra de herramientas "Show all Events" para ver todos los eventos. Para usar un evento que no es visible, usted necesitará revisar las propiedades de evento en el archivo de recurso.

Para crear un nuevo archivo GUIBuilder. (gbf) haga lo siguiente:

1 Use una de las siguientes opciones:

· En el menú archivo, seleccione Nuevo.

· En el toolbar, pulse el botón el botón Nuevo.

· Press <Ctrl>+ N.

2 El cuadro de diálogo de nuevo nombre de archivo GUIBuilder aparece. Muévase al directorio en el que usted quiere guardar el archivo, entre un nombre de archivo (ningún sufijo del archivo es necesario) y pulse el botón guardar.

3 El cuadro de diálogo para crear un recurso aparece. Use una de las siguientes opciones:

· Si usted no ha creado un archivo de recurso, y quiere llamar al ResBuilder para crear uno, pulse el botón Sí. (Al terminar ResBuilder, usted volverá a GUIBuilder.)

· Si usted ya ha creado un archivo de recurso, y quiere continuar creando el archivo .gbf, pulse el botón No.

· Si usted quiere permanecer en ResBuilder, pero descontinuar el nuevo proceso de creación de archivo, pulse el botón Cancelar.

4 El diálogo de abrir un archivo de Recurso aparece y lo posiciona para seleccionar un archivo de recurso. Muévase al directorio que contiene el archivo de recurso, selecciónelo, y pulse el botón abrir.

5 El diálogo de opciones de programa aparece para permitirle definir las siguientes opciones para el programa a ser creado:

· para especificar una ruta de búsqueda, digite la ruta en el campo de PREFIX de búsqueda.

· para especificar el número de dígitos de precisión numérica, digite el valor deseado (el rango aceptable es -1 a 16, donde -1 indica el máximo de precisión) en el campo de precisión.

· para especificar el número de páginas a usar para un chequeo de memoria mínimo, digite el número en campo de páginas de chequeo de memoria mínimo.

· para especificar cuales formas se despliegan en el inicio del programa, pulse uno de los botones siguientes: Muestre todo el form(s) al inicio del programa, Muestre sólo la primera forma al inicio del programa, o no muestra ninguna forma al inicio del programa.

· para incluir comentarios en el programa, pulse el check-boton Incluya comentarios en programa. Los comentarios siempre son incluidos en la versión fuente del programa generado.

· para incluir un mensaje que aparezca en el principio del código del programa, digite un string de texto en el campo de Mensaje de Derechos de propiedad literaria.

6 Digite la información deseada y pulse el botón OK para continuar. (La información de las opciones puede modificarse en cualquier momento vía el comando de las Opciones del Programa.)

7 El nombre del archivo GUIBuilder es desplegado en la barra de título.

Abriendo archivos existentes de GUIBuilder

Para abrir un archivo (.gbf) GUIBuilder existente, haga lo siguiente:

1 Si el archivo se lista en la lista de MRU (archivos recientemente usados), selecciónelo. Por otra parte, use una de las siguientes opciones para desplegar el cuadro de dialogo de File GUIBuilder que le permite seleccionar el archivo:

· En el menú File, seleccione Open.

· En el toolbar, pulse el botón el botón Open.

· Press <Ctrl>+ O.

2 El cuadro de dialogo de File GUIBuilder aparece. Muévase al directorio que contiene el archivo .gbf que usted quiere abrir, seleccione el archivo, y pulse el botón Abrir.

Imprimiendo Archivos

Para imprimir el archivo GUIBuilder (.gbf), haga lo siguiente:

1. Despliegue el cuadro de diálogo de Impresión por cualquiera de las opciones siguientes:

· En el menú Archivo, Print.
· En el toolbar, pulse el botón de Impresión.
· Press <Ctrl>+P.

2. Seleccione las opciones de impresión deseadas y pulse el botón OK.

Nota: Si no hay ningún dispositivo SYSPRINT definido en el archivo config.bbx, esta opción no estará disponible.

Guardando Archivos

El actual archivo GUIBuilder (.gbf) es automáticamente guardado siempre que usted salga de GUIBuilder, cierre el archivo .gbf, o abra un nuevo archivo .gbf. También se salva cada vez que usted Ve, Construye, o Ejecuta el programa generado.
Para guardar el archivo en cualquier otro momento, use una de las siguientes opciones:

· En el menú del Archivo, Save.
· En el toolbar, pulse el botón Save.
· Press <Ctrl>+ S.

Trabajando con bloques de Código
Definiendo Bloques de Código

Definiendo Eventos en Bloques de Código

Definiendo un bloque de código involucra definir el código de Visual PRO/5 a ser ejecutado en la ocurrencia del evento. Hay dos maneras de seleccionar el bloque del código a ser revisado:

· vía lista desplegable (Listbox)
· pulsando el botón o objeto en la forma a la que el evento será atado.

Método de Lista (ListBox)

1. Pulse el botón del Listbox de la lista Object para desplegar las formas en el archivo de recurso. (La lista Object también despliega el código de Inicialización, Fin del Trabajo, y definiciones de Subprogramas y Funciones, pero nosotros no estamos a estas alturas interesados en ellos.) Seleccione la forma que contiene el recurso (forma, ventana, o control) al que usted quiere definirle el bloque de código.

2. Si el recurso es un Child Window, o se encuentra dentro de un Child Window, pulse el botón de listbox de la lista Window y seleccione el Child Window. (La lista de Window muestra la forma seleccionada inicialmente en la lista de Object.) Si no hay ningun Child Window en la forma seleccionada, la propia forma ya se seleccionará y se desplegará en la lista de Window, y se deshabilitará para que no pueda editarse.

3. Pulse el botón de listbox de la lista Control para desplegar la lista de controles o objetos contenida en la forma o child window que seleccionó en la lista Window. Para definir un bloque de código para la forma o child window que seleccionó en la lista de Window, selecciónelo de nuevo en la lista Control. Si la forma seleccionada o child window no contiene ningún objeto, ya se seleccionará en la lista Control, y se deshabilitará para que no pueda ser editado

4. Pulse el botón de Listbox de Evento y seleccione un evento. Si hay sólo un evento disponible para el objeto seleccionado en la lista Control, ya se seleccionará en la lista de Evento, y se deshabilitará para que no pueda ser editado. El código del evento se inicializará con un bloque de comentario para identificar el evento; para algunos eventos, se pondrán también variables del objeto para ayudarlo.
Si el evento no es visible, un cuadro de diálogo de advertencia del Evento se despliega. Éste lo advierte de que usted necesitará revisar los atributos de la forma en el archivo de recurso para hacer este evento visible dentro del loop de eventos.

5. Si usted está usando un editor externo, pulse el botón Edit Code de la barra de herramienta para revisar el bloque de código que usa su editor externo.

6. Digite o edite el código de Visual PRO/5 a ser ejecutado en el evento.

Método visual

1. Use una de las siguientes opciones:

· En el menú Program, seleccione Select Form, entonces seleccione la forma deseada del submenu-alterno.
· En la barra de herramientas, pulse el botón Select Form, entonces seleccione la forma deseada en el cuadro de diálogo de selección de formas y pulse el botón OK.

2. Si GUIBuilder se configura para proporcionar ayuda extra en este punto, el cuadro de diálogo de selección de la forma se despliega para proveer instrucciones para trabajar con la forma. (Puede evitar que este diálogo se despliegue cada vez que usted selecciona una forma, pulse el checkbox no me muestre de nuevo esta pantalla) Pulse el botón OK para salir del cuadro de selección de la Forma.

3. La forma seleccionada aparece. Use una de las siguientes opciones:
· para definir un evento en conjunto para la forma, doble-clic, en el fondo de la forma.
· para definir un evento para un objeto específico, doble-clic en ese objeto o control.
· para definir un evento para una opción de un menú, seleccione ese ítem del menú.
· para definir un evento para el close box, de click en el close box.
Para salir de la forma sin definir ningún evento, presione la tecla "Del".

4. Si usted da doble click en una parte de la forma en donde existe más de un control (uno sobre otro), el diálogo de seleccionar control aparece. Seleccione un control de la lista y pulse el botón OK.

5. Para seleccionar un evento:
· Si usted seleccionara un control que sólo apoya un tipo de evento, o si usted está definiendo un evento para un ítem del menú o la caja de cierre, el evento correcto se selecciona automáticamente.
· Si usted dio doble click en el fondo de la forma o en un control que apoya más de un tipo de evento, el diálogo de selección de evento aparece. Seleccione un evento de la lista y pulse el botón OK.

6. Incluso cuando un control apoya un cierto tipo de evento, que el evento no podría ser visible en el loop de evento del programa generado. Esto significa que el evento es optativo, y no se ha habilitado para esta ventana en ResBuilder. Si un evento semejante es seleccionado, el cuadro de dialogo de advertencia de evento aparece. Este diálogo indica que, mientras usted puede proseguir y puede crear el código para el evento, no será visible en el loop de eventos a menos que usted regrese al ResBuilder y revise las propiedades de la ventana y habilite el evento. (Los eventos siguientes siempre son visibles: se operó la caja de cierre; se seleccionó un ítem del menú; Un push botón fue presionado; Un tool button fue presionado; y cualquiera que Notifique eventos, del tipo de código “N”).

7. Si usted está creando un nuevo evento, el bloque de código se inicializa con un comentario para identificar el evento. Para algunos eventos, se ponen también variables del control para ayudarlo. Después de que el bloque del comentario inicial fue insertado por GUIBuilder, entre su propio código de Visual PRO/5 a ser ejecutado siempre que este evento ocurra en el programa generado. No hay necesidad de salvar cada bloque de código específicamente; cuando usted sigue al próximo bloque de código, o cuando usted cierra el archivo o termina el programa, el bloque de código se salva automáticamente a su archivo .gbf.

Inicialización
Fin del Trabajo
Subrutinas y Funciones

Definiendo el Código de Inicialización

Para definir el código a ser ejecutado inmediatamente antes del loop de eventos (como abrir archivos de datos, definir plantillas de registro (templates), inicializar variables globales, almacenar cajas de lista "listbox"), haga lo siguiente:

1. Use una de las siguientes opciones:
   · En el menú Program, Initialization.
   · Seleccione en la barra de herramientas, clic en el botón Edit Subroutine, entonces una caja de lista de selección aparece.
   · Click o flecha abajo de la caja de lista de Objeto y seleccione Initializacion Code.
2. Digite el código de Visual PRO/5 a ser ejecutado inmediatamente antes de entrar en el loop de eventos.

Definiendo el Código de fin del Trabajo

Para definir el código a ser ejecutado inmediatamente después del loop de eventos (como cerrar archivos de datos o realizar cualquier otra limpieza que fuera necesaria), haga lo siguiente:

1. Use una de las siguientes opciones:
   · En el menú Program, seleccione End of Job.
   · En la barra de herramientas, pulse el botón Edit Subroutine, entonces seleccione End of Job de la lista.
   · Click o flecha abajo en la caja de lista de Object y seleccione End of Job Code.

2. Digite el código a ser ejecutado inmediatamente después de terminar el loop de eventos.

Definiendo Subrutinas y Funciones

Para editar código en subrutinas, funciones, o declaraciones no-ejecutables como TABLE, IOLIST, y DATA, haga lo siguiente:

1. Use una de las siguientes opciones:
· En el menú Program, Subroutines/Functions.
· Click en la flecha abajo de la caja de lista de Object y Subroutines/Functions.
· Seleccione en la barra de botones, pulse el botón Edit Subroutine, entonces seleccione una subrutina de la lista, o Nuevo Subroutine/Function para crear una nueva subrutina o función.

2. Si usted está definiendo una nueva subrutina o función, el diálogo "Nombre Subroutine/Function" aparece. Digite un nombre descriptivo para esta subrutina o función y pulse el botón OK.
Para definir una función, digite una función sola o multi-línea siguiendo las reglas de Visual Pro/5.
Para definir una subrutina digite una única etiqueta finalizada con dos puntos ":", seguida por el código de Visual PRO/5 que lleva a cabo la subrutina, y finalmente un RETURN.· para definir una declaración de no-ejecución, entre en una única etiqueta de la declaración que acaba con dos puntos, seguida por un TABLE, IOLIST, o el verbo DATA de acuerdo a las reglas de Visual PRO/5. Usted puede referirse a esta declaración en otra parte del programa usando la etiqueta (TBL=ETIQUETA, IOL=ETIQUETA, o RESTORE ETIQUETA; DREAD Data-list, GOSUB ETIQUETA).

Verificando Errores en Bloques de Código

Para verificar los errores en el bloque de código actual:

1. Use una de las siguientes opciones:
· En el menú de Program, seleccione Check for Errors.
· En la barra de botones, pulse el botón Check for Errors.

2. Uno de los siguientes diálogos es desplegado:
· Si ningún error se encuentra en el bloque actual de código, se despliega el mensaje "Ningún error encontrado."
· Si se encuentran uno o más errores en el bloque actual de código, el diálogo de "Lista de Errores" es desplegado con una lista de mensajes de error y muestra el número de la línea relativo dentro del bloque actual para cada error .
a) Seleccione uno de los errores de la lista, entonces apriete el button close.
b) El "focus" es transferido a la ventana de edición de Código. La línea del error también será seleccionada.

· La función de construcción de Programa del menú Program verifica los errores, pero sólo informa un error en un momento. El chequeo de errores incluido con la función de construir programa "Build Program" es más comprensivo, porque puede examinar el programa entero en lugar de simplemente una línea de código a la vez.

· Esto es un parámetro en el archivo gb.ini llamado check_syntax. Si este parámetro se pone a Yes, GUIBuilder verifica los errores para cada bloque de código cuando se salva el archivo .gbf. En cualquier error encontrado, el diálogo de "Syntax Errors" se desplegará con el mensaje de advertencia "Este código tiene errores de sintaxis." (Nota: La función del check_syntax se pone a NO por defecto; para habilitarlo, revise el archivo gb.ini.)

Eliminando Bloques de Código

Para borrar el bloque actual de código, haga lo siguiente:

1. Despliegue el bloque del código a ser borrado, entonces use una de las siguientes opciones:
· En el menú Program, seleccione Delete Code.
· En la barra de botones, pulse el botón "Delete current code block".
2. Un cuadro de diálogo de confirmación de borrado aparece y pide confirmación de borrado. Pulse el botón Yes para confirmar el borrado o No para cancelar el borrado.

Consiguiendo Código Externo

Para cargar un programa externo o archivo de texto, haga lo siguiente:

1. En el menú Program, seleccione "Get External Code"

2. El diálogo Abrir Programa de Visual PRO/5 o Archivo Fuente aparece.
Digite el nombre de un archivo de texto o un programa de Visual PRO/5, Si usted digita el nombre de un archivo de texto, este es cargado en la Ventana de Vista y el enfoque "focus" se transfiere a la Ventana de Vista. Si usted carga un archivo de programa de Visual PRO/5, el proceso siguiente toma lugar:
a) En una copia temporal del programa, todas las referencias de número de línea se convierten en etiquetas, y se crean tantas etiquetas de línea como sea necesario.
b) La copia temporal del programa se lista a un archivo de texto que usa el parámetro lister del archivo gb.ini. Éste normalmente será pro5lst.exe.
c) La versión listada del archivo de programa de Visual PRO/5 está cargada en la Ventana de Vista y el enfoque se transfiere a la Ventana de Vista.

3. Ahora que la lista del programa esta en la Ventana de Vista, usted puede hacer lo siguiente:

· Para navegar dentro de la lista del programa, use las teclas PGUP/PGDN o las flechas arriba/abajo
· para copiar porciones del programa listado en la ventana de Edición de Código, haga lo siguiente:

a) Seleccione uno o más líneas.
b) Copie las líneas seleccionadas al portapapeles usando la opción de copiar del menú.
c) Cambie a la Ventana de Edición de Código seleccionando "Window" y 0 Edit Code del menú o pulsando el botón en la barra de botones "Toggle Window (Edit/View)".
d) Y finalmente, seleccione la opción Pegar (Paste) del menú.

Usted puede pasar de un lado a otro entre la ventana de Edición de Código (Ventana 0) y la Ventana de Vista (Ventana 1) usando la opción "Window" Ventana 0 y Ventana 1 en el menú o pulsando el botón "Toggle Window (Edit/View)" en la barra de botones. El programa listado en la Ventana de Vista (Ventana 1) permanecerá hasta la próxima vez usted ejecute una de las siguientes acciones:
· Recibir Código Externo de Programa "Get ExternalCode" del menú Program.
· View Program del menú Program.

Usando un Editor Externo

La conducta predefinida del GUIBuilder es permitir editar en la ventana de “Edit Code”. Para usar un editor de texto externo, haga lo siguiente:

1. Antes de empezar GUIBuilder, asegurese de que la línea "editor=" en el archivo **/basis/tools/guibuild/gb.ini** apunte al editor externo especificado.
2. Cuando GUIBuilder se empieza y un bloque de código se especifica, el botón de herramienta de Editor Externo y Editor Externo en el menú "Tools" se habilita.
3. Operando el botón o seleccionando el ítem del menú causará que GUIBuilder desactive la ventana "Edit Code", escriba el bloque del código actual a un archivo temporal, y invoque al editor para editar el bloque de código en el archivo temporal.
4. Guarde este archivo temporal en formato ASCII. Terminadores de línea CR/LF son soportados.
5. Cuando usted termina el editor externo, GUIBuilder inserta el bloque del código modificado en el archivo .gbf.

Debido a que los archivos .gbf están en formato ASCII, pueden ser vistos con cualquier editor en cualquier momento, pero es recomendable que para modificar su código, que se haga solo a través del GUIBuilder. Aunque los archivos .gbf incluyen código fuente BBx, ellos no son la fuente del archivo y tienen un formato propio. Si el formato de un archivo GUIBuilder está perdido o dañado, podría resultar totalmente inutilizable o podría llevar a resultados imprevisibles.

Si le interesa un buen editor externo, puede usar el ‘editplus.exe’, el cual puede conseguir por Internet en la dirección www.editplus.com por unos $30.

Verificando Recursos

Para verificar un archivo GUIBuilder (.gbf) contra el archivo de recurso correspondiente, haga lo siguiente:

1. En el menú Program, seleccione Check Resource.

2. Si ningún error se encuentra, el siguiente mensaje se despliega: “Este archivo GUIBuilder es consistente con el archivo de recurso 'nombre del archivo'. Ningún error se encontró.”
3. Si cualquier error se encuentra en el archivo de recurso, el diálogo Error en Archivo de Recurso se despliega, con uno de los siguientes mensajes:

a) Los siguientes mensajes indican que el archivo de recurso se ha vuelto corrupto:

Missing [Program] block.
Multiple [Init] blocks.
Multiple [EOJ] blocks.
Missing ‘Program Name' variable
Missing ‘Resource File' variable.
Invalid Event header : [Event…]·
Error en programa _qres, Enumerate_Res_Forms, usando filename de archivo de recurso.
Error en programa _qres, Enumerate_Res_Child_Windows, usando filename de archivo de recurso.
Error en programa _qres, Enumerate_Res_Controls, usando filename de archivo de recurso.

b) El mensaje siguiente indica que el archivo de recurso se ha vuelto corrupto o no está disponible:
'No se puede abrir el archivo de recurso filename'.
c) Los mensajes siguientes indican que el archivo .gbf se refiere a estructuras de datos que ya no existen en el archivo del recurso:
Ventana ID X ya no existe en el archivo de recurso.
Control ID X no existe en ventana ID Y.
Event Code X no está disponible para el control ID Y en ventana ID Z.
Junto con uno de los mensajes anteriores, GUIBuilder pregunta “Anula la rutina de evento(s) del archivo .gbf?”
Seleccione una estas alternativas:
Sí - Prosigue y anula la rutina(s) del archivo .gbf.
No - Retenga la rutina(s) en el archivo .gbf.
Cancelar - Detenga el proceso de verificación en el recurso.

Verificando las Opciones del Programa

Cuando un archivo .gbf se crea, GUIBuilder proporciona la capacidad para especificar opciones para el programa a ser generado. Estas opciones pueden modificarse como sigue:

1. En el menú Program, seleccione Program Options.
2. El diálogo de Opciones de Programa aparece y le permite definir las opciones siguientes para el programa a ser creado:

Para especificar un PREFIX de ruta de búsqueda de archivos, digite la ruta en el campo de PREFIX de búsqueda.

Para especificar el número de dígitos de precisión numérica, digite el valor deseado (el rango aceptable es -1 a 16) en el campo de precisión

Para especificar el número de páginas de memoria mínimo, digite el número de páginas en el campo de Memory check pages.

Para especificar qué formas se desplegarán al inicio del programa, pulse en uno de los siguientes radiobuttons: Muestre todo el form(s) al inicio del programa, Muestre sólo la primera forma al inicio del programa, o no muestra ninguna forma al inicio del programa.

Para incluir comentarios en el programa, pulse el radiobutton “Incluir comentarios en programa”.

Para incluir un mensaje que aparezca cerca del principio del código del programa, digite un String de texto en el campo de Derechos de propiedad literaria (Copyright Message).

3.Click OK para cerrar el diálogo de Opciones de Programa.

Construyendo Programas

Para construir un programa, haga lo siguiente:

1. Use una de las siguientes opciones:
· En el menú Program, seleccione "Build Program".
· En la barra de botones, pulse el botón "Build Program".
2. El diálogo de 'Guardar programa como' aparece. Digite el nombre del programa.

Si el programa ya existe, el diálogo de advertencia “Guardar Programa como” se despliega, con el mensaje “el Programa ya existe. Borra el archivo existente?” Seleccione 'Yes' para proseguir y borrar el archivo existente, o 'No' para entrar un nombre de programa diferente.

3. El programa completo se ensambla en un archivo fuente según las reglas descritas en la rutina de programa gb_func::gb__build_program.

4. El archivo fuente generado es tokenized (compilado) usando el programa identificado por el parámetro del tokenizer del archivo gb.ini (normalmente pro5cpl.exe).
Si cualquier error es detectado por el compilador, el diálogo “Build Program Error” se despliega, con el mensaje “el compilador reporta un error en línea N de la rutina X: error-message.” Pulse el botón OK para reconocer el error.
GUIBuilder llama la rutina con el error y posiciona el cursor en la línea que esta con el error.

Si ningún error es descubierto por el compilador, el diálogo “GUIBuilder” se despliega, con el mensaje “Program 'nombre del programa' se construyó con éxito.” presione el botón OK

Viendo Programas

Para ver el programa, haga lo siguiente:

1. En el menú Program, seleccione View Program
2. Si el programa necesita ser reconstruido, el proceso de construcción de programa se invoca en este punto.
3. cuando el programa se ha construido con éxito, la lista del programa fuente está cargado en la Ventana de Vista. Use PGUP/PGDN y las flechas arriba/abajo para navegar dentro del programa listado.
4. usted puede trasladarse de un lado a otro entre la ventana Edit Code (Ventana 0) y la Ventana de Vista (Ventana 1), seleccionando del menú la opción Window y Ventana 0 o Ventana 1 o pulsando el botón en la barra de botones "Toggle Window (Edit/View)". El programa permanecerá en la Ventana de Vista (Ventana 1) hasta la próxima vez que usted seleccione uno de las siguientes opciones:
"Get External Code" del menú Program.
"View Program" del menú Program.
"Read Me" del menú de Ayuda.

Corriendo Programas

Para ejecutar programa compilado, haga lo siguiente:

1. Use una de las siguientes opciones:
. En el menú Program, seleccione "Run Program".
. En la barra de botones, pulse el botón "Run Program".
2. Si el programa necesita ser reconstruido, el proceso de construcción de programa se invoca en este punto.
3. Cuando el programa se ha construido con éxito, GUIBuilder lo ejecuta en una copia secundaria del interprete Visual PRO/5 y usa el parámetro intérprete del archivo gb.ini. Por defecto, cuando el programa generado termina, termina la consola de Visual PRO/5; digite BYE o RELEASE para volver a GUIBuilder.

Preparando programas para el usuario final

El programa y el archivo de recurso son requeridos para cada programa; también transporte el archivo .gbf si usted piensa hacer desarrollo en el sitio en cada caso, GUIBuilder debe instalarse en el lugar del usuario final. El programa _qres siempre debe existir en el sitio del usuario final.

Archivo	**Descripción**
program.brc or *program*.brf	Resource file –debe ser creado primero, usando Resbuilder (.brc), ResEdit (.brf), or ResCompiler (.arc compiled to .brc). Requerido en runtime.
program.gbf	Inicialmente creado basado en el archivo *program*.brc; modificado por el desarrollador usando GUIBuilder.
program.src	Generado desde *program*.brc, *program*.gbf, gb_*.cod, y las reglas internas del GUIBuilder.
program.bbx	Compilado desde *program*.src usando pro5cpl.exe. Note que el usuario puede cambiar la extensión cambiando el parámetro program_extension= en el archivo gb.ini. Requerido en runtime.
program.err	Temporalmente, creado y borrado cuando se construye el programa.
_qres	Query Resource Functions (Utilitario). Requerido en runtime.

Tópicos Avanzados

Conversiones de Carácter-GUI usando Procedimientos por Lote (Batch)

La base de su código existente es su recurso más grande, pero también es su barrera más grande moviendose a un ambiente GUI.
El Ambiente de Programación Visual Guibuilder ofrece un poco de ayuda con la habilidad de cortar-y-pegar código de los programas existentes en un nuevo programa GUI.
Pero si usted tiene mucha entrada de datos en el programa y le gustaría poner todo junto en un procedimiento de actualización por lote (batch), usted necesitará caminar fuera del Ambiente de Programación Visual.
Si la base de su código existente y esquemas de pantalla están suficientemente bién estructurados, usted podría parcialmente automatizar el proceso de conversión. Esto es posible porque los componentes que constituyen GUIBuilder se llevan a cabo como llamadas a procesos independientemente.

Aquí esta una misma apreciación global del proceso:

1. Escriba un programa para leer la estructura actual de su pantalla (de archivos de pantalla o directamente del programa fuente) y escriba archivos de recurso ASCII. Vea "ResCompiler: Otro Camino al Diccionario de Interface de Usuario" en <http://www.basis.com/advantage/mag-v2n1/rescompiler.html>.

Una manera de simplificar este proceso es crear su archivo ASCII de Recurso con la unidad de medida definida como "Caracteres" (lo opuesto a "Pixeles" o "Semichars"). Porque las pantallas para ambiente de caractéres no trazan directamente a las pantallas de GUI, usted necesitará tomar algunas decisiones sobre cómo convertir varios elementos de la pantalla. La manera más fácil de empezar es convertir texto estático al control TEXT, campos string a controles INPUTE, campos numéricos a controles INPUTN, y las cajas y líneas a GROUPBOX y controles de LÍNEA. Para una discusión más completa de este proceso, vea "A New Approach to Going GUI" en <http://www.basis.com/advantage/mag-v1n3/goinggui.html>.

2. Compile cada archivo de recurso ASCII (sample.arc) a un archivo binario de Recurso (sample.brc) usando programa ResCompiler (rescomp.exe).

3. Genere el armazón para un archivo control GUIBuilder (sample.gbf) llamando el módulo de GUIBuilder "gb_func::gb__make_gbf."

4. Escriba un programa para analizar su programa y extractos que pueden usarse en una versión de GUI del programa. Para ayudar con este proceso, usted podría usar el programa "_label" que convierte todas las referencias de número de línea en un programa que hace referencia a líneas de etiqueta, y el utilitario pro5lst.exe que convierte un archivo de programa a texto. Cuando usted extrae estos pedazos, escríbalos en el archivo de control de GUIBuilder (sample.gbf) usando la función fngb__put_code$() incluida en gb_func.src. Piense en términos de las siguientes áreas funcionales:

Inicialización. Apertura de Archivos; definición de estructuras de registro (templates) y otras variables globales. Note que GUIBuilder genera una línea ENTER con una sola variable (gb__arg$) que puede tener un template asociado para permitir valores múltiples de paso. El código de inicialización se escribe en el archivo control de GUIBuilder con el encabezado de [Init] .

Fin de Trabajo. Cerrar Archivos; limpiar variables globales.
El fin de Trabajo "End of Job" es escrito en GUIBuilder con el encabezado de [EOJ].

Eventos. Éste es el más difícil, porque aquí es donde las estructuras en caractér y programas GUI son muy diferentes. Como una aproximación muy áspera, considere usar los eventos "Got Focus" y "Lost Focus", consiguió focus y perdió focus.

"Got Focus": Este evento está disponible para todos los campos de entrada de datos (INPUTE, INPUTN, TEXT, y TXEDIT). Se activa cuando el usuario entra en el campo.

"Lost Focus": También disponible para todos los campos de entrada de datos, este evento se activa cuando el usuario deja un campo. No asuma que el usuario necesariamente termina con un campo sólo porque usted consiguió el evento "Lost Focus". Podría significar (por ejemplo) que el usuario apenas llamó al sistema de ayuda para verificar algo.

En un programa GUI, la manera más fiable para los usuarios de informarlo de que ellos terminan con una pantalla dada es presionar un botón "OK". Si usted tiene un marco de entrada de datos normal, usted podría querer considerar generar una serie de botones automáticamente para manejar funciones normales (Actualizar Registro, Borrar Registro, Imprimir Registro, etc.).

Subprogramas, Funciones, y bloques de código no-ejecutable.
Éstos pueden ser relativamente fáciles de pasar al archivo control de GUIBuilder, pero depende de usted asegurar que ellos se están usando apropiadamente. Los subprogramas deben incluir una etiqueta y una declaración de retorno "RETURN", y bloques de código no-ejecuble (IOLIST, DATOS, y MESA) debe tener una etiqueta asociada para que usted pueda referirse a ellos en otra parte del programa. Éstos bloques de código se escriben en GUIBuilder con un encabezado de [Función (nombre_único)]. Si usted tiene subrutinas estandar, funciones, IOLISTs, TABLEs, o otros bloques de código que debe ser incluido en todos sus programas, póngalos en el archivo estandar gb_std.cod. Este archivo se une en el final de todos los programas generados por GUIBuilder. Los mensajes estandar de error y mensajes de escape son puestos en los archivos de texto ASCII (gb_err.cod y gb_esc.cod); si es necesario, usted puede reemplazarlos con mensajes personalizados de error y manejadores de escape.

5. Confirme que la estructura del archivo de control GUIBuilder (sample.gbf) acepta llamadas del módulo de GUIBuilder "gb_func::gb__val_data."

6. Genere el final del programa (sample.bbx) llamando el módulo de GUIBuilder "gb_func::gb__build_program"

7. En este punto, el programa GUI generado puede mantenerse dentro del ambiente de programación visual GUIBuilder.

Archivo de Parámetros gb.ini

Lo siguiente identifica los parámetros del archivo gb.ini.

Parámetro	**Valor Inicial**	**Descripción**
about_box_win	-1,-1	Posición X,Y del cuadro de diálogo

		"About GUIBuilder".
bitmap_dir		Dice al GUIBuilder donde buscar los archivos bmp usados en la barra de herramientas.
check_syntax	No	Si está puesto, GUIBuilder chequeará la sintaxis cada bloque de código antes de salvar en la estructura del archivo .gbf. El usuario puede cheques sintaxis en cualquier momento con la selección en el menu de *Program->Check for Errors*.
config_bbx	..\..\config.min	Ruta del archivo de configuración config.bbx (usado por *Tools->VPRO/5* y por *Program->Run*).
copyright		Si está puesto, esto se copiará a la línea de Copyright= en el archivo .gbf, el cual causará que se ponga un REM al principio del programa generado.
ddbuilder	..\..\vpro5\ddbuild.exe	Programa DDBuilder (debe ser ddbuild.exe).
editor		Si esta puesto, GUIBuilder llamará este editor externo en lugar de usar el control TXEDIT para la construcción de programas.
error_list_win	-1,-1	Posición X,Y de la ventana de lista de error usada para reporte de errores de sintaxis al usuario.
event_list_win	-1,-1	X,Y location of the event list window used to prompt the user to pick an event for a selected control.
find_win	-1,-1	Posición X,Y de la ventana "find text" (buscar texto).
help_file	guibuild.hlp	Nombre del archivo de ayuda de GUIBuilder.
include_comments	Yes	Si está puesto, esto copiará la línea Remarks= (Comentarios=) en el archivo .gbf, el cual determina si el programa compilado tendrá remarks, con la bandera -r (omite comentarios).
interpreter	..\..\vpro5\vpro5.exe	Intérprete Visual PRO/5.

Intro_video_file		Nombre del archivo, si existe, este desplegará un video introductorio de guía didáctica de GUIBuilder.
Lister	..\..\vpro5\pro5list.exe	Nombre del programa para listar de Visual PRO/5.
main_win	-1,-1	Posición X,Y[,W,H] de la ventana principal del GUIBuilder.
memory_check_pages		Si está puesto, esto copiará la línea Pages= en el archivo .gbf, el cuál causará que se realice un test del mínimo de memoria requerido para ser puesto al principio de los programas generados.
new_program_win	-1,-1	Posición X,Y de la pantalla de ayuda que describe como crear un nuevo programa usando GUIBuilder.
options_win	-1,-1	Posición X,Y de la ventana (*Program->Options*).
other_code_win	-1,-1	Posición X,Y de la ventana "Other Code".
pick_event_help_w	-1,-1	Posición X,Y de la ventana de “Pick in Events” (Seleccionar Eventos)
precision		Si esta puesto, esto copiará la línea Precision= en el archivo .gbf, el cual generará la línea PRECISION al principio de los programas generados. (Rango - 1..16)
prefix		Si está puesto, esto copiará la línea Prefix= en el archivo .gbf, el cual generará una línea PREFIX al principio de los programas generados.
program_extension	bbx	La extensión para los programas generados. Si no se pone (program_extension=) es permitido, y causará que los programas compilados sean creados con extensión. Archivos fuente tienen extensión .src.
recent_files	4	Numero de archivos .gbf que se mantienen en la lista (MRU) Most Recently Used (más recientemente usados) menu (range 0..9).
recent_file1.. recent_file9		Archivos .gbf más recientemente usados (se actualiza cuando cada archivo .gbf es

		abierto y cuando este es guardado).
Release	Yes	Si esta puesto GUIBuilder saldrá de Visual PRO/5, retornando a Windows cuando usted salga.
Resbuilder	..\..\vpro5\resbuild.exe	Nombre del programa editor de recurso (Usualmente RESBUILD.EXE, pero podría ser RESEDIT.EXE).
select_control_win	-1,-1	Posición X,Y de la ventana "Select Control", para seleccionar controles que se sobreponen en la misma posición en la pantalla.
show_all_events	Yes	Si esta puesto, la caja de lista de Evento desplegará todos los eventos, aún si ellos no son visibles en el loop de eventos del programa generado
select_form_win	-1,-1	Posición X,Y de la ventana "Select Form"
show_forms	All	Esto se copiará a la línea 'Show Forms=' en el archivo .gbf, el cual determina como se inicializan las ventanas en el programa generado. Valores permitidos con ninguno o (0): inicialmente esconde todas las ventanas; First o (1): inicialmente esconde todas las ventanas excepto la primera encontrada en el archivo de recurso; y All o (2), el default: inicialmente muestra todas las ventanas.
show_pick_event_help	Yes	Determina si el usuario verá el diálogo de explicación acerca de como atar código a un evento determinado.
show_welcome_window	Yes	Determina si el usuario verá la pantalla de bienvenida inicial cuando empieza GUIBuilder. El usuario puede hacer click en el checkbox sobre la ventana de bienvenida y suprimirla en el futuro
splash_screen_win	-1,-1	Posición X,Y de la pantalla de inicial copyright que aparece cuando empieza GUIBuilder.
subroutine_win	-1,-1	Posición X,Y de la ventana para escribir el nombre de una nueva subrutina o función.
tokenizer	..\..\vpro5\pro5cpl.exe	Nombre del programa compilador calificado de Visual PRO/5.

user_carriage_returns	No	Si está puesto, GUIBuilder creará ciertos archivos de salida usando terminadores de línea CRLF, el default es solo LF).
welcome_win	-1,-1	Posición X,Y de la ventana de bienvenida de GUIBuilder.

Formato del archivo .gbf

Cuando usted define un nuevo programa en GUIBuilder, se guarda la información que relaciona al nuevo programa en un archivo de control GUIBuilder, con una extensión '.gbf '. Este archivo de trabajo .gbf trabaja en conjunto con un archivo de recurso de BASIS que debe crearse primero. Pueden crearse archivos de recurso de BASIS en ResEdit (Visual PRO/5 v.1 formato .brf), ResBuilder (Visual PRO/5 v.2 formato .brc), o cualquier editor de texto (Visual PRO/5 v.2 formato .arc). Archivos de Recurso ASCII (formato .arc) debe convertirse a formato .brc antes de que ellos puedan ser usados por GUIBuilder. Esta conversión puede hacerse dentro de ResBuilder (Archivo->Open->Recurso ASCII), o usted puede usar el programa ResCompiler (rescomp.exe).

Esta sección identifica el formato del archivo .gbf y lista las variables en el [Programa] sección del archivo

Secciones de Archivo

El archivo de control de GUIBuilder es una archivo de texto con formato ASCII con una serie de secciones cada una de los cuales son identificadas por una ficha en llaves cuadradas [] empezando al margen izquierdo. El formato global del archivo es:

Sección Archivo	**Descripción**
[Program]	variable name = value
[Init]	el código de inicialización de Programa va aquí.
[EOJ]	el final del programa y códigos de limpieza va aquí.
[Remark]	los comentarios Optativos van aquí; ellos son ignorados por GUIBuilder.
[Event …]	Las secciones de manejo de eventos de estructuran como sigue: [Event Win=***window_id*** ID=***control_id*** Code=***event_code*** <***event_name***> (***subroutine_name***)] ***window-id*** identifica la forma o child window dentro del archivo de recurso. Una forma principal con un ID de 10 aparecería como Win=10. Un child window con un ID de 1001 dentro de una forma con un ID de 10 aparecería como Win=10.1001. Un child window de 2001 dentro de la ventana ID 10.1001 aparecerían como Win=10.1001.2001.

control_id es un número asignado a un control en particular en el archivo de recurso. Está en el rango de 1 a 32767. Control ID 0 indica la propia ventana; se identifican eventos de la ventana con ID=0.

Event_code es una sola letra o dígito (ej. 'B'=button pushed, 'X'=window close) o una letra seguida por un digito entero (ej. 'f0'=lost focus, 'f1'=focus gained). Para Notificar eventos, ':'+tipo-control se añade al código de evento. Por ejemplo, "N2:19" (se hizo la selección de botón de lista), "N24:107" (se insertó una fila en un grid). Cada tipo de control apoya a diferentes juegos de códigos de evento.

Event_name es una descripción corta del código de evento. Por ejemplo, Code=X tendría un nombre de evento <WIN_CLOSE>.

Subroutine_name es el nombre internamente-generado del evento de la subrutina dentro del programa generado. El ejemplo siguiente demuestra el formato usual de un nombre de subrutina:

[Event Win=101.1002 ID=1001 Code=f0 <FOCUS_LOST>
(**W101_1002_C1001_FOCUS_LOST**)]

Si GUIBuilder genera un nombre de subrutina de evento que excede 32 carácteres, se le indicará al usuario que digite un nuevo nombre de subrutina.

[Program] Sección de Variables

Lo siguiente ilustra las variables en la sección [Program] del archivo .gbf. La mayoría de estos valores se ponen fijos en el diálogo de Opciones de Programa.

Nombre de Variable	**Valor de Ejemplo**	**Descripción**
Resource File	myprog.brc	Nombre del archivo de recurso usado para crear el archivo .gbf. la ruta del directorio nunca es dado; el archivo de recurso debe estar en el directorio actual, o debe estar localizable usando el prefix de Visual PRO/5.
Program Name	myprog	Nombre base del programa, sin extensión especificada. El programa fuente es escrito a myprog.src; la extensión para el programa compilado esta en el parámetro program_extension del archivo gb.ini; el

		default es bbx.
Remarks	Yes	Determina si REMs (comentarios) son copiados al programa compilado. REMs siempre aparecen en la versión .src
Creation Date	1998-06-15	Fecha de creación del archivo .gbf
Creation Time	11:20:43	Hora de creación del archivo .gbf
Copyright	Copyright (C) 1998 BASIS International Ltd.	Si esta puesto, esto causa que el copyright especificado se inserte cerca del inicio del programa generado.
Show Forms	All	Determina la visibilidad inicial de las formas en el programa generado. Los posibles valores son “All” (hace todas las formas visibles), “First” (hace la primer forma visible, pero inicialmente esconde cualquier otra), o “None” (empieza con todas las formas visibles; depende del desarrollador cuando ellas se harán visibles).
Prefix	c:/data/ d:/data/ c:/prog/	Si está puesto, esto causa que la declaración de PREFIX sea creada cerca del principio del programa generado.
Precision	4	Si está puesto, esto causa que la declaración de PRECISION sea creada cerca del principio del programa generado. El rango es de 0..16, o -1 para el punto flotante.
Pages	1024	Si está puesto, esto causa que un test de memoria sea insertado cerca del principio del programa generado. Si el programa no puede empezar con por lo menos este parámetro de páginas, este empezará con más memoria. Note que este parámetro se ignora si el programa es llamado 'CALL'.

Variables del Programa generadas

Ésta es una lista de las variables y funciones disponible a diseñadores en programas generados por GUIBuilder.

Variables disponibles en todas las partes del programa (incluso la Inicialización y fin del programa):

Variable	Descripción
gb__arg$	Argumento pasado en una declaración CALL y devuelto en la línea ENTER del programa generado.
gb__args	Numero de valores pasados en gb__arg$. gb__args podría ser -1 si gb__arg$ es una variable simple (sin un template), 0 si el programa fue corrido (o si el programa fue llamado y gb__arg$ no fue pasado), o un entero positivo para indicar el número de campos definidos en el template asociado con gb__arg$.
gb__sysgui$	Nombre del alias SYSGUI, tomado del archivo config.bbx
gb__sysgui	Canal en el cual el dispositivo SYSGUI (gb__sysgui$) es abierto
gb__sysprint$	Nombre del alias SYSPRINT, si uno fue definido en el config.bbx. Si ningún dispositivo SYSPRINT existe en el config.bbx, el gb__sysprint$ será nulo ("").
gb__win_id$	Inicialización: El primer ID de ventana encontrado en el archivo de recurso. End of Job: el último ID de ventana referenciado en el Loop de Eventos.
gb__win$	Template asociado con el string conteniendo cada forma y nombre de ventana definido en el archivo de recurso del programa; el valor de cada campo es el contexto de la forma dada o ventana. Por ejemplo si una forma se nombra “Main” y el contexto es 0, entonces gb__win.main será 0.

Variables Disponible Sólo dentro del loop de eventos

Variable	Description
gb__event$	Registro de Evento. Este es dimensionado usando TMPL(gb__sysgui). El registro de evento esta documentado en la guía GUI de Visual PRO/5.
gb__notice$	El string de aviso para la notificación actual de evento. Este string es solo significativo si gb__event.code$ es "N.” Será dimensionado con el template correcto basado en la notificación de evento.
gb__win_id$	Ventana actual ID. Por ejemplo, si la ventana actual es una forma principal con un ID de 100, entonces el gb__win_id$ es "100". Si la ventana actual es un child window con un ID de 1001 en una forma principal de 100, entonces el gb__win_id$ es "100.1001" pueden anidarse child windows a un nivel arbitrario, y es posible ver gb__win_id$ como "100.1111.2222.3333."
gb__eoj	Esta variable puede ponerse a cualquier valor que no sea cero para forzar la terminación del loop de eventos.

Ambos gb__event$ y gb__notice$ son fijos en el loop de eventos:

```
gb__event$:tmpl(gb__sysgui)
gb__event=len(gb__event$)
dim gb__generic$:noticetpl(0,0)
gb__eoj=0

repeat
  read record (gb__sysgui,siz=gb__event,err=gb__event_loop_end)gb__event$
  if gb__event.code$="N" then
:    gb__generic$=notice(gb__sysgui,gb__event.x%);
:    gb__notice$:noticetpl(gb__generic.objtype%,gb__event.flags%);
:    gb__notice$=gb__generic$


  cuerpo del loop de evento

until gb__eoj
```

Note que todas las variables definidas empiezan con gb__ (con dos rayas "underscores"). Semejantemente, todas las funciones definidas empiezan con fngb__.

_label

Convierte Números de Línea a Líneas de Etiquetas

Sintaxis

call “_label”,inpgm$,outpgm$,errmsg$,always_create

Descripción

Esta utilidad examina un programa de Visual PRO/5 para todas las referencias de número de línea, las convierte a líneas que referencian una etiqueta, y genera etiquetas de línea como sea necesario. La salida se escribe a un archivo de programa separado. Esta utilidad es usada por el GUIBuilder en la función "Get Existing Code".

Argumento	**Descripción**
inpgm$	Nombre del programa fuente. No se modificará de forma alguna.
outpgm$	Programa de Salida (destino). Si existe, se borrará.
errmsg$	Retornado como "" si el outpgm$ se creó ok. Retornado como "OK" si ninguna conversión fue necesaria y outpgm$ no fue creado.

	Retornado como un mensaje de error si un problema fue encontrado.
always_create	Si es 0: No crea el outpgm$ si no hay línea de etiqueta a convertir. Si es 1: Crea el outpgm$ aún cuando no hay ninguna línea de etiqueta para convertir.

gb_rec

Copia Campos Entre Registros

Sintaxis

call “gb_rec”,gb__rec1$,gb__rec2$,gb__skip$,gb__copied$,gb__skipped$,gb__rec_err

Descripción

Este programa se usa para copiar datos de un registro a otro en una base de campo-por-campo. Los datos se copian basados en campos con nombres idénticos; los tipos de los datos son más insignificantes, y los datos se convertirán cuando sea necesario.

Datos que no podrían convertirse (ej. que copia “X” a un campo numérico) se informará en la lista gb__skipped$.

Argumento	**Descripción**
gb__rec1$	Campos son copiados desde aquí (registro fuente)
gb__rec2$	Campos son copiados aquí (registro destino)
gb__skip$	Se usa el delimitador Linefeed para crear una lista de nombres de campo a saltar en la copia (optativo)
gb__copied$	Retorna una lista delimitada con linefeed de cada nombre de campo que se copió
gb__skipped$	Retorna una lista delimitada con linefeed de cada nombre de campo que se saltó
gb__rec_err	0 = éxito -1 = gb__rec1$ no tiene un template -2 = gb__rec2$ no tiene un template

Funciones para Leer y Actualizar la Pantalla

(fngb__template$, fngb__get_screen$, fngb__put_screen$)

Estas tres funciones le permiten que trabaje con controles que usan los nombres que asignó en el ResBuilder (opuesto al uso de ID´s numéricos para controles) los cuales proporcionan una manera simplificada de manipular controles gráficos.

Function	**Description**
fngb__template$()	Retorna un template (plantilla) que describe los controles en una

	ventana específica.
fngb__get_screen$()	Copia el valor de cada uno de los controles de la pantalla al registro de la plantilla (template).
fngb__put_screen$()	Actualiza cada uno de los controles de la pantalla basado en los valores del registro del template.

Estas funciones proporcionan toda la funcionalidad que usted necesitará para la mayoría de las situaciones, pero en algunos casos, usted necesitará usar mnemónicos más precisos, así como las funciones CTRL() y SENDMSG(). Por ejemplo, estas funciones leen o escriben listas completas de o para botones de lista y controles de caja de lista. Para determinar el ítem actualmente seleccionado en una caja de la lista o botón de lista, use la función CTRL(); para cambiar el ítem actualmente seleccionado o ítems, use uno de los mnemónicos 'LISTSEL ', 'LISTMSEL ', o 'LISTUNSEL'.

La siguiente tabla muestra como cada tipo de control mapea a la estructura de datos de plantilla:

Control Type	**Template Format**	**Value Description**	**Get Value: Screen -> String**	**Put Value: String -> Screen**
Push Button (11)	C(1*=0)	N/A	N/A	N/A
Radio Button (12)	N(1*=0)	0=unchecked 1=checked	dec(ctrl(gb__sysgui, gb__ctl_id,2))	'CHECK'
Check Box (13)	N(1*=0)	0=unchecked 1=checked	dec(ctrl(gb__sysgui, gb__ctl_id,2))	'CHECK' or 'UNCHECK'
Horizontal Scrollbar (14)	N(4*=0)	scrollbar position based on the 'scrollrange'	dec(ctrl(gb__sysgui, gb__ctl_id,2))	'SCROLLPOS'(gb__ctl_id,int)
Vertical Scrollbar (15)	N(4*=0)	scrollbar position based on the 'scrollrange'	dec(ctrl(gb__sysgui, gb__ctl_id,2))	'SCROLLPOS'(gb__ctl_id,int)
Edit (16)	C(64*=0)	Text value of the control.	ctrl(gb__sysgui, gb__ctl_id,1)	'TITLE'
Static Text (17)	C(64*=0)	Text value of the control.	ctrl(gb__sysgui, gb__ctl_id,1)	'TITLE'
List Box (18)	C(255*=0)	Complete list, with ítems delimited by line feeds.	ctrl(gb__sysgui, gb__ctl_id,7)	'LISTSUSPEND' 'LISTCLR' 'LISTADD' 'LISTRESUME'
List Button (19)	C(255*=0)	Complete list, with ítems delimited by line feeds.	ctrl(gb__sysgui, gb__ctl_id,7)	'LISTSUSPEND' 'LISTCLR' 'LISTADD' 'LISTRESUME'

List Edit (20)	C(64*=0)	Text value of the edit portion of the control.	ctrl(gb__sysgui, gb__ctl_id,1)	'TITLE'
Group Box (21)	C(1*=0)	N/A	N/A	N/A
Custom Edit (22)	C(64*=0)	Text value of the control.	ctrl(gb__sysgui, gb__ctl_id,7)	'TITLE'
Menu Ítem (100)	N(1*=0)	0=unchecked 1=checked	dec(ctrl(gb__sysgui, gb__ctl_id,2))	'CHECK' or 'UNCHECK'
Checkable Menu Ítem (101)	N(1*=0)	0=unchecked 1=checked	dec(ctrl(gb__sysgui, gb__ctl_id,2))	'CHECK' or 'UNCHECK'
Status Bar (102)	C(64*=0)	Text value of the control.	ctrl(gb__sysgui, gb__ctl_id,1)	'TITLE'
Tool Button (103)	N(1*=0)	0=unchecked 1=checked	dec(ctrl(gb__sysgui, gb__ctl_id,2))	'CHECK' or 'UNCHECK'
INPUTE (104)	C(64*=0)	Text value of the control.	ctrl(gb__sysgui, gb__ctl_id,1)	'TITLE'
INPUTN (105)	C(16*=0)	Text value of the control (use NUM() to convert to numeric).	ctrl(gb__sysgui, gb__ctl_id,1)	'TITLE'
Tab (106)	N(4*=0)	Currently selected tab index.	dec(sendmsg (gb__sysgui, gb__ctl_id,29,0,$$))	sendmsg(gb__sysgui, gb__ctl_id,34,0,$$)
Grid (107)	C(1*=0)	N/A	N/A	N/A
Line (108)	C(1*=0)	N/A	N/A	N/A
Image (109)	C(1*=0)	N/A	N/A	N/A

fngb__template$(gb__win_id$)

Obtener String Template por medio del ID de un contexto.

Descripción

Esta función devuelve el TEMPLATE de un contexto principal. La plantilla contiene un campo para cada control en la forma especificada o ventana, con los tipos de los datos y significados listados anteriormente en la tabla. El siguiente ejemplo es basado en el programa "Hello" descrito en "Getting Started":

```
>dim Hello$:fngb__template$(gb__win_id$)
>print fattr(Hello$,"")
STATUS
PUSH_ME
MESSAGE
```

```
>print fattr(Hello$)

STATUS:C(64*=0):ID=1 TYPE=102 X=0 Y=178 W=320 H=22:,
PUSH_ME:C(1*=0):ID=1001 TYPE=11 X=110 Y=50 W=90 H=25:,
MESSAGE:C(64*=0):ID=1002 TYPE=17 X=50 Y=110 W=200 H=25:
```

Se asignan nombres de campo basado en los nombres del ResBuilder. Si el nombre en ResBuilder está en blanco, el nombre de campo de plantilla será "ID" + el str(ctl_id). Al definir la plantilla, si GUIBuilder encuentra un nombre de campo que ya reproduce uno en la plantilla, agregue "_+str(ctl_id) al extremo del segundo nombre del campo para hacerlo único. Si el nombre del campo en ResBuilder contiene cualquier carácter que es ilegal en los identificadores de Visual PRO/5 (ej. espacios), ellos se convertirán a underscores. Por ejemplo, un nombre de campo en ResBuilder de "Over Credit Limit?" aparecería en la plantilla como "OVER_CREDIT_LIMIT". Los carácteres " " y "?" se convierte a underscores

Campo	**Descripción**
ID	Control ID asignado en ResBuilder. Usted puede obtener el valor ID del template usando fattr(rec$,fld$,"ID"). Por ejemplo, fattr(Hello$,"Message","ID") devuelve "1002".
TYPE	Tipo de campo numérico como se definió en la tabla anterior. Usted puede obtener el valor de TYPE desde el template usando fattr(rec$,fld$,"TYPE"). Por ejemplo, fattr(Hello$,"Message","TYPE") devuelve "17", que indica que este campo es un control de texto estático.
X, Y, W, and H	Posición del control en la pantalla (X=posición horizontal en pixeles, Y=posición vertical en pixeles, W=ancho en pixels, y H=alto en pixeles). Usted puede obtener cualquiera de estos valores desde el template usando fattr(rec$,fld$,*parameter*). Por ejemplo, fattr(Hello$,"Message","X") devuelve "50", que indica que empieza en la posición horixontal pixel 50.

fngb__get_screen$(gb__win_id$,screen $)

Copia el valor del control de la Pantalla al Registro de la Plantilla

Descripción

Esta función recupera datos del screen/window en una variable string que usted dimensiona usando la función fngb__template$(). Aquí es un ejemplo del programa "Hello":

```
>Hello$=fngb__get_screen$(gb__win_id$,Hello$)
>print Hello.Status$

>print Hello.Push_Me$

>print Hello.Message$
¡Hola de GUIBuilder!
```

fngb__put_screen$(gb__win_id$,screen$)

Actualizar los valores de controles de Pantalla del Registro de la Plantilla

Descripción

Esta función toma datos de la variable string que usted dimensionó usando la función fngb__template$() y actualiza al screen/window. Aquí es un ejemplo del programa “Hello”

```
>Hello.Status$= "This is a status message"
>Hello.Message$= "Hello, GUIBuilder"
>Hello$=fngb__put_screen$(gb__win_id$,Hello$)
```

Funciones para Recuperar Información de la Ventana

(fngb__win_id$, fngb__context, fngb__win_info$)

fngb__focus_win_id

Recibe ID de la Ventana del Contexto Actual.

Uso

```
gb__win_id$=fngb__win_id$(gb__event.context)
```

fngb__context

Recibe el Contexto de la Ventana ID

Uso

```
gb__context=fngb__context(gb__win_id$)
```

fngb__focus_winl_info

Consigue el Bloque de Información de la Ventana para una Ventana ID dada.

Uso
gb__win_info$=fngb__win_info$(gb__win_id $)

Plantilla de Información de la Ventana

```
dim gb__win_info$:"class:u(1),type:u(1),hidden:u(1),disabled:u(1),"
:                 +"context:u(2),eventmask:u(4),flags:u(4),focus:u(2),"
:                 +"x:i(2),y:i(2),w:u(2),h:u(2),title:c(16*=)"
```

El bloque de información de la ventana se contruye utilizando las siguientes funciones de Visual Pro/5

```
    gb__win_info$=ctrl(gb__sysgui,0,4,gb__context)
:             +ctrl(gb__sysgui,0,8,gb__context)
:             +bin(gb__context,2)
:             +sendmsg(gb__sysgui,0,21,0,$$,gb__context)
:             +sendmsg(gb__sysgui,0,22,0,$$,gb__context)
:             +ctrl(gb__sysgui,0,2,gb__context)
:             +ctrl(gb__sysgui,0,0,gb__context)
:             +ctrl(gb__sysgui,0,1,gb__context)
```

Funciones para Poner Enfoque de la Pantalla

(fngb__focus_win_id, fngb__focus_ctl_id, fngb__focus_ctl_name)

Estas funciones se usan para poner enfoque a un window/screen en particular, y opcionalmente a un control en particular.

fngb__focus_win_id

Ponga Enfoque a una Ventana ID dada.

Uso
gb__context=fngb__focus_win_id(gb__win_id$)

fngb__focus_ctl_id

Ponga Enfoque a una Ventana ID dada y control ID

Uso
gb__context=fngb__focus_ctl_id(gb__win_id$,gb__ctl_id)

fngb__focus_ctl_name

Poner Enfoque hacia una Ventana de un ID y nombre del control indicados.

Uso
gb__context=fngb__focus_ctl_name(gb__win_id$, "Push_Me")

Función para obtener el Template de un Child Window o de un Tab

Ejemplo, en el que “chw_ventana1” es el nombre dado al control.

Dim frm_screen$:fngb__template$(fngb__win_id$((nfield(gb__win$,”chw_ventana1”)))

Print (gb__sysgui)’context’(gb__win.chw_ventana1)

Función SENDMSG() – Enviar Mensaje a Windows y Controles

Sintaxis

```
SENDMSG(sysgui,id,function,int,string${,context{,ERR=lineref}})
```

Descripción

La función SENDMSG () envía los mensajes a las ventanas y los controles. El significado del string retornado depende del tipo de objeto del parámetro de identificación referido y el mensaje a ser enviado.

Mientras la función SENDMSG () contenida en esta sección aplica a ventanas y ventanas hijas, existen las funciones SENDMSG () que aplican específicamente a los controles Grid, INPUTE, INPUTN, y TAB, así como también todos aplican específicamente al uso de teclados internacionales. Lo siguiente identifica los parámetros comunes a todas las funciones SENDMSG ():

Parámetro	Descripción
Sysgui	Un número de canal válido para SYSGUI.
Id	Identificación del Objeto. Para operaciones del sistema, usar un valor de -1. Usar un valor de 0 cuando se esté refiriendo a la ventana del contexto en uso. Cuando parámetro del contexto es incluido, usando un valor de 0, se refiere a la ventana de el contexto especificado.
function	Número de la función a ser usada (ver tabla en siguiente página).
Int	Entero de cuatro dígitos que contienen un valor necesario para la función indicada.
string$	Argumento para la función indicada.
context	Contexto que contiene el ID especificado. Este parámetro solo es necesario para objetos que no pertenecen al contexto en uso.
ERR=lineref	No.de línea o Etiqueta a donde se bifurca si un error ocurriera durante la ejecución.

Lista de Funciones en orden numérico

20 Obtener/Establecer texto, color, alineamiento y estilo de una celda.
21 Establecer valor para una o varias celdas continuas en una sola columna (Ver Ej).
22 Establecer valor para una determinada celda (Ver Ej).
23 Poner títulos de encabezados.
24 Establecer el ancho de las columnas con base en parámetros pasados (Ver Ej).
25 Iniciar operación de arrastrar y quitar datos.
26 Finalizar modo de edición.
27 Establecer opciones extendidas para texto (solo para revisión 2.0x).
28 Establecer el espacio entre el Grid y los encabezados de columnas y líneas.
29 Ajustar el ancho de las columnas indicadas rellenando el cuerpo entero del Grid.
30 Establecer características de un Grid con características de otro Grid (Ver Ej).
31 Iniciar modo de edición en una determinada celda.
32 Des-seleccionar todas las celdas del Grid.
33 Indica si el Grid acepta funciones de arrastre desde otro Grid.
34 Retorna el valor editado en una celda que tiene un control INPUTE.
35 Establecer el valor a ser editado en una celda, sin que sea desplegado.
36 Establecer el ancho de determinada columna en el Grid.
37 Obtener string binario con el ancho promedio de un carácter, basado en font usado.
38 Obtener string binario con el ancho de una columna en pixeles.
39 Obtener información del font utilizado en el Grid.
40 Obtener string binario con el número de columnas en el Grid.
41 Obtener string binario con el número de líneas en el Grid.
42 Obtener string binario con el alto de determinada línea, en pixeles.
43 Obtener número máximo de columnas que pueden ser desplegadas en el Grid.
44 Obtener string binario con número de la columna seleccionada.
45 Obtener string binario con número de la línea seleccionada.
46 Obtener string binario con número de línea en el tope de la ventana del Grid.
47 Especificación de una columna como la columna en uso.
48 Especificación de una línea como la línea en uso.
49 Poner celdas y líneas en modo resaltado.
50 Refrescar determinada línea.
51 Establecer efecto tridimensional de sombra en el encabezado del Grid.
52 Establecer color para sombra oscurecida en el encabezado del Grid.
53 Establecer color para sombra clara en el encabezado del Grid.
54 Establecer características en una celda y mostrar datos en esta o en encabezado (Ej).
55 Establecer color de fondo para el Grid.
56 Establecer color para el texto en el Grid.
57 Establecer método por el cual las celdas serán puestas como resaltadas.
58 Establecer el despliegue de líneas horizontales entre líneas en el cuerpo del Grid.
59 Establecer cómo una columna en el Grid será scroleada.
60 Establecer el inter-espacio entre líneas, levantando el alto total de las celdas.
61 Establecer el color de la separación entre líneas del Grid especificado.
62 Establecer el margen izquierdo y derecho para texto en el Grid principal.
63 Establecer el modo de margen a aplicar, a la derecha o izquierda.

64 Establecer el modo de capturar el mouse para Grids principales.
65 Establecer el comportamiento del mouse al hacer clicks y arrastres.
66 Establecer el número de columnas en el Grid especificado (Ver Ej).
67 Establecer el número de líneas en el Grid especificado (Ver Ej).
68 Establecer el alto de las líneas del Grid principal, sin incluir el encabezado (Ver Ej).
69 Establecer barra de scroll vertical, des-habilitando modo de actualización.
70 Despliega determinada línea por el tope de la ventana del Grid.
71 Establecer modo de scroll vertical para el Grid.
72 Controlar el despliegue de líneas verticales entre columnas en el cuerpo del Grid.
73 Obtener string binario con el ancho de determinado string, basado en el font usado.
74 Establecer si el usuario podrá cambiar el ancho de las columnas con el mouse.
75 Junto con función 54 permite el despliegue de una imagen en una celda.
76 Establecer refrescamiento automático.
77 Saber/Establecer si encabezados aparecen en modo oprimido.
78 Establecer el cursor usado para operaciones de arrastrar/eliminar información.
79 Establecer la máscara de edición para una determinada columna.
80 Establecer el medio a usar para atar el Grid a un canal abierto (Ver Ej).
81 Ejecutar una de cinco operaciones con un Grid atado a un canal.
82 Establecer opción para cuando no existe llave primaria y hay un nuevo registro.
83 Establecer el estilo de celda para el Grid (Ver Ej).
84 Establecer la forma de alinear los datos en las celdas del Grid (izq, der, cen).
85 Obtener y desplegar los datos del último registro del canal atado al Grid.
86 Lo mismo de la función 85, pero mostrando una caja de diálogo.
87 Establecer el valor para una o más celdas continuas en una sola línea (Ver Ej).

Lista de Funciones agrupadas por funcionalidad

Dimensión de Grids

66 Establecer el número de columnas en el Grid especificado (Ver Ej).
67 Establecer el número de líneas en el Grid especificado (Ver Ej).

Dimensión de Columnas y Líneas

24 Establecer el ancho de las columnas con base en parámetros pasados (Ver Ej).
29 Ajustar el ancho de las columnas indicadas rellenando el cuerpo entero del Grid.
36 Establecer el ancho de determinada columna en el Grid.
60 Establecer el inter-espacio entre líneas, levantando el alto total de las celdas.
68 Establecer el alto de las líneas del Grid principal, sin incluir el encabezado (Ver Ej).

Apariencia de las Celdas

55 Establecer color de fondo para el Grid.
56 Establecer color para el texto ene el Grid.

61 Establecer el color de la separación entre líneas del Grid especificado.
83 Establecer el estilo de celda para el Grid (Ver Ej).
84 Establecer la forma de alinear los datos en las celdas del Grid (izq, der, cen).

Separación de Líneas

58 Establecer el despliegue de líneas horizontales entre líneas en el cuerpo del Grid.
72 Controlar el despliegue de líneas verticales entre columnas en el cuerpo del Grid.

Márgenes

62 Establecer el margen izquierdo y derecho para texto en el Grid principal.
63 Establecer el modo de margen a aplicar, a la derecha o izquierda.

Resaltado

49 Poner celdas y líneas en modo resaltado.
57 Establecer método por el cual las celdas serán puestas como resaltadas.

Funcionalidad

33 Indica si el Grid acepta funciones de arrastre desde otro Grid.
59 Establecer cómo una columna en el Grid será scroleada.
65 Establecer el comportamiento del mouse al hacer clicks y arrastres.
69 Establecer barra de scroll vertical, des-habilitando modo de actualización.
71 Establecer modo de scroll vertical para el Grid.
74 Establecer si el usuario podrá cambiar el ancho de las columnas con el mouse.
75 Junto con función 54 permite el despliegue de una imagen en una celda.
78 Establecer el cursor usado para operaciones de arrastrar/eliminar información.
79 Establecer la máscara de edición para una determinada columna.

Características con Múltiples Grids

30 Establecer características de un Grid con características de otro Grid (Ver Ej).

Características de las Celdas

27 Establecer opciones extendidas para texto (solo para revisión 2.0x).
54 Establecer características en una celda y mostrar datos en esta o en encabezado (Ej).

Consultas

20 Obtener/Establecer texto, color, alineamiento y estilo de una celda.
34 Retorna el valor editado en una celda que tiene un control INPUTE.
37 Obtener string binario con el ancho promedio de un carácter, basado en font usado.

38 Obtener string binario con el ancho de una columna en pixeles.
39 Obtener información del font utilizado en el Grid.
40 Obtener string binario con el número de columnas en el Grid.
41 Obtener string binario con el número de líneas en el Grid.
42 Obtener string binario con el alto de determinada línea, en pixeles.
43 Obtener número máximo de columnas que pueden ser desplegadas en el Grid.
44 Obtener string binario con número de la columna seleccionada.
45 Obtener string binario con número de la línea seleccionada.
46 Obtener string binario con número de línea en el tope de la ventana del Grid.
73 Obtener string binario con el ancho de determinado string, basado en el font usado.

Despliegue del Grid

47 Especificación de una columna como la columna en uso.
48 Especificación de una línea como la línea en uso.
50 Refrescar determinada línea.
70 Despliega determinada línea por el tope de la ventana del Grid.
76 Establecer refrescamiento automático.

Insertar Datos

21 Establecer valor para una o varias celdas continuas en una sola columna (Ver Ej).
22 Establecer valor para una determinada celda (Ver Pgm).
35 Establecer el valor a ser editado en una celda, sin que sea desplegado.
54 Establecer características en una celda y mostrar datos en esta o en encabezado (Ej).
87 Establecer el valor para una o más celdas continuas en una sola línea (Ver Ej).

Operaciones con el Grid

25 Iniciar operación de arrastrar y quitar datos.
25 Finalizar modo de edición.
31 Iniciar modo de edición en una determinada celda.
32 Des-seleccionar todas las celdas del Grid.
64 Establecer el modo de capturar el mouse para Grids principales.

Encabezados del Grid

23 Poner títulos de encabezados.
28 Establecer el espacio entre el Grid y los encabezados de columnas y líneas.
51 Establecer efecto tridimensional de sombra en el encabezado del Grid.
52 Establecer color para sombra oscurecida en el encabezado del Grid.
53 Establecer color para sombra clara en el encabezado del Grid.
77 Saber/Establecer si encabezados aparecen en modo oprimido.

Grids Atados a un Archivo

80 Establecer el medio a usar para atar el Grid a un canal abierto (Ver Ej).
81 Ejecutar una de cinco operaciones con un Grid atado a un canal.
82 Establecer opción para cuando no existe llave primaria y hay un nuevo registro.
85 Obtener y desplegar los datos del último registro del canal atado al Grid.
86 Lo mismo de la función 85, pero mostrando una caja de diálogo.

GUIBuilder: La Manera Fácil de Ir hacia GUI

Las interfases gráficas para el usuario han revolucionado la manera en que los usuarios interactúan con el software. En vez de mecanografiar comandos memorizados, ahora pueden señalar simplemente, para poder realizar sus operaciones más complejas. El resultado ha hecho la computación mucho más fácil para el promedio de usuarios pero ha creado un paso de aprendizaje para los desarrolladores.

Los programas GUI son estructuralmente diferentes de los programas para ambiente de caracteres, con la pantalla transformada en una ventana gráfica y campos de datos reemplazados por controles gráficos como los botones de pulsar, las cajas de chequeo, y los botones de radio. Tenemos entonces el loop de eventos, que es el corazón del programa manejador de eventos del programa GUI, esperando en ciclos a que el usuario tome alguna acción – sea oprimir un botón o hacer clic en un menú – lo cual es detectado por el loop de eventos. Y para que sea aún menos complicado, las ventanas y los controles gráficos se pueden identificar con nombres fáciles-de-recordar en vez de números arbitrarios.

Esta nueva herramienta que se ofrece a partir de Visual PRO/5® REV.2.0 ha simplificado la creación de manejadores de eventos en programas GUI para los programadores que ya tengan algún nivel de experiencia. GUIBuilder™ es un ambiente visual de programación completo que hace que los desarrolladores enfoquen el diseño del programa en los que realmente es importante – diseñando sobre el flujo de la aplicación y creando las reglas del negocio – automáticamente manejando todos los eventos que nos detalla el controlador de eventos del programa GUI. GUIBuilder incluye una serie de funciones definidas que permiten que usted trabaje los controles gráficos con nombres significativos, en lugar de los números arbitrarios que antes se usaban. Para desarrolladores que puedan haber visto el ambiente GUI como algo intimidante y excesivamente complejo, GUIBuilder hace que la programación GUI sea más fácil y entendible.

El Ambiente de Programación Visual

El primer paso para desarrollar un programa con GUIBuilder no comienza verdaderamente con GUIBuilder si no con ResBuilder™, el nuevo editor de pantallas gráficas que trae el Visual PRO/5 desde la revisión 2.0. Usando ResBuilder, usted crea una ventana -- un tipo de forma de GUI -- y entonces coloca los controles gráficos en la ventana. ResBuilder guarda la estructura de la ventana y sus controles en un archivo de recursos. Hecho eso, luego usted desde GUIBuilder puede abrir el archivo de recursos e iniciar escogiendo las formas, ventanas, controles, y eventos de listas, y digitar en *manejadores de eventos* -- bloques con código de Visual PRO/5 que indicarán al GUIBuilder cómo responder a los eventos seleccionados. Cuando usted ha terminado la creación de las subrutinas *manejadoras de eventos*, GUIBuilder genera un completo y extensamente documentado programa GUI.

La orientación visual de GUIBuilder elimina la posibilidad de errores en la identificación de formas, ventanas, controles, eventos al permitirle escoger los valores correctos desde listas. Para asegurar que el programa permanezca al día con cambios al archivo de recursos, un detallado chequeo cruzado es hecho cada vez que usted carga el programa en GUIBuilder y cuando usted retorna desde la edición del archivo de recursos en GUIBuilder.

GUIBuilder no puede convertir automáticamente todos sus programas basados en caracteres a GUI, pero lo ayuda a incorporar el código existente basado en caracteres en sus nuevos programas para GUI. La interfase del GUIBuilder tiene dos ventanas, un Editor Window para la digitación de código en bloques y un Visor Window, donde usted puede cargar los archivos secundarios de texto o programas. Cuando un programa tokenizado de Visual PRO/5 se carga en el Visor Window, GUIBuilder convierte todas las referencias de números la línea del programa en etiquetas y carga el programa con un formato de texto. Usted entonces puede cortar fácilmente los bloques escogidos de código de su programa heredado y los pega en el programa nuevo de GUI, intercambiando entre el Editor Window y el Visor Window.

El chequeo de error instantáneo que verifica la sintaxis con el BBx tradicional® en modo de consola no se pierde. GUIBuilder ofrece una opción de menú y el botón correspondiente que le permiten verificar la sintaxis en cualquier momento. Esta característica verifica la sintaxis del bloque de código actual y regresa un mensaje "No errores encontrados" o una lista de errores. Cuando un error de la lista es escogido, el cursor se coloca en una ventana de edición con la línea de código erróneo en un modo sobresaltado. Al final, cuando el programa es compilado, una última verificación detallada de la sintaxis es realizada para informarle si quedó algo malo. Otra opción puede ser usada para verificar y obtener automáticamente una lista de los errores de sintaxis, cuando salve cada bloque de código al archivo de trabajo.

Además de las características arriba descritas, GUIBuilder le permite:

- Crear o editar un archivo de recursos en ResBuilder, mientras esté trabajando con GUIBuilder en sincronización con el archivo de recursos.
- Trabajar opcionalmente con un diccionario de datos en DDBuilder™.
- Abrir una sesión separada en modo de consola de Visual PRO/5 para que trate de probar la sintaxis en forma interactiva.

Estructura del Programa que se genera

Con GUIBuilder, las partes que usted no ve -- la armazón y la lógica asociada a la generación de código -- son apenas tan importantes como las partes que usted ve. En GUIBuilder, los bloques de código que usted crea son insertados en una armazón uniforme que guarda el control del detalle relativo a los manejadores de eventos del programa GUI. La estructura del programa:

- Abre y cierra el dispositivo SYSGUI.
- Carga todos los recursos (formas, ventanas, y controles) desde archivo de recursos y le asigna un número único de contexto a cada ventana.
- Construye el loop de eventos que chequea la cola de eventos e invoca las rutinas manejadoras de eventos a como se necesiten.
- Crea los TEMPLATES (de la forma) para que usted pueda trabajar con nombres de control en lugar de números de control.
- Genera el código necesario para tratar en forma normal las interrupciones por ESCAPE y control de errores.
- Junta todas las piezas para generar un programa completo.

GUIBuilder genera un programa completo y de trabajo aún antes de que usted agregue sus propios bloques de código. Usted puede sentarse solo o con un cliente y diseñar la pantalla gráfica completa con el ResBuilder e inmediatamente generar un prototipo de programa que demuestra la belleza de la aplicación y experimente. Usted puede incluso levantar el programa generado del sitio del cliente. Cuando combine con el archivo de recursos y un simple programa invocado por CALL, **`_qres`**, el cual es requerido en tiempo de corrida, el programa generado es completamente independiente del ambiente de GUIBuilder. Si tuviera curiosidad acerca de la estructura completa del programa, sepa que los programas generados por GUIBuilder quedan muy bien documentados y legibles.

Construcción de Bloques GUI

Un programa creado por medio de GUIBuilder se compone de una estructura uniforme, más bloques de código individual. Los bloques de código se catalogan en una de las siguientes categorías:

- **Inicialización.** Este tipo de bloque de código incluye todo lo que usted requiere hacer al principio del programa, inmediatamente antes del loop de eventos. Típicamente aquí, usted puede abrir los archivos de datos, definir TEMPLATES, e inicializar las variables normales y globales que vaya a necesitar.
- **Manejador de eventos.** Manejadores de eventos son subrutinas individuales invocadas a como son requeridos en el loop de eventos. Como programador, usted no puede llamar las rutinas para tratamiento de eventos. Usted tendrá que decidir lo que quiere hacer cuando un evento es dado, por ejemplo, como cuando un botón específico es empujado.
- **Fin del Trabajo.** Esto incluye todo lo que usted quiere hacer al final del proceso, inmediatamente después de que el loop de eventos termine. Por lo general, usted cerraría los archivos de datos, limpiaría variables, o bien, podría realizar el retorno hacia un menú.

Hay otro tipo de bloques con código, conocido como Subrutinas o Funciones, que son usadas por subrutinas comunes, funciones definidas, o código no ejecutable como el de sentencias TABLE, IOLIST o DATA. Código definido en estos bloques no es manejado automáticamente en la estructura del programa, así que usted tendrá que decidir cómo usarlo.

Hay unas pocas características más interesantes asociadas con la creación de las rutinas del manejador de eventos. Cuando usted crea a un nuevo manejador de eventos, el bloque de código no empieza completamente en blanco. Este es inicializado con suficiente código para que usted inicie, a veces con una línea de comentarios como un **`REM Button was pushed`**. Para eventos más complejos, el bloque se inicializa con una serie de líneas de comentario y otro código para permitir que usted sepa qué variables del control están disponibles y las varias opciones asociadas con el control. Cuando usted crea a un nuevo manejador de eventos, GUIBuilder también chequea la máscara del evento para la ventana escogida, advirtiendo si la máscara de eventos necesita ser cambiada para hacer el evento visible en el programa generado.

Funciones para Manejo de Datos

Los lenguajes de programación vinieron a proporcionar mucho más facilidad cuando los programadores pudieron utilizar nombres de variables descriptivas como **`product_id$`** y **`customer_name$`** en lugar de **`p1$`** y **`n2$.`** En la misma forma, GUIBuilder permite que usted trabaje con los mismos controles que usan los nombres definidos en el ResBuilder en lugar de los números identificadores del control (ID), logrando con esto que el código que escriba sea mucho más entendible. Usted no necesita preocuparse por números de contexto y números de control directamente, ya que estos son manejados automáticamente por las funciones para manejo de datos.

Si hasta ahora usted no ha logrado programar con alguna herramienta GUI, ahora es el momento de probar. Programar en ambiente GUI con GUIBuilder es lograr que el desarrollo de programas sea intuitivo y a través de ayudas del sistema, con solo eso se le hace más fácil comenzar. Si usted tiene ya alguna experiencia de escribir programas con el manejador de eventos GUI de la revisión 1.0x de Visual PRO/5, usted quedará impresionado con cuán rápido y fácil ha llegado a ser esta labor con GUIBuilder.

Lo que realmente se necesita para que usted entre al Diseño con GUI

El aspecto más desafiante de programar con GUI no es la mecánica de tener que manejar ventanas y controles. Muchos de los detalles mecánicos se pueden automatizar. La verdadera dificultad está en el aprendizaje de cómo tener que pensar en una manera enteramente nueva.

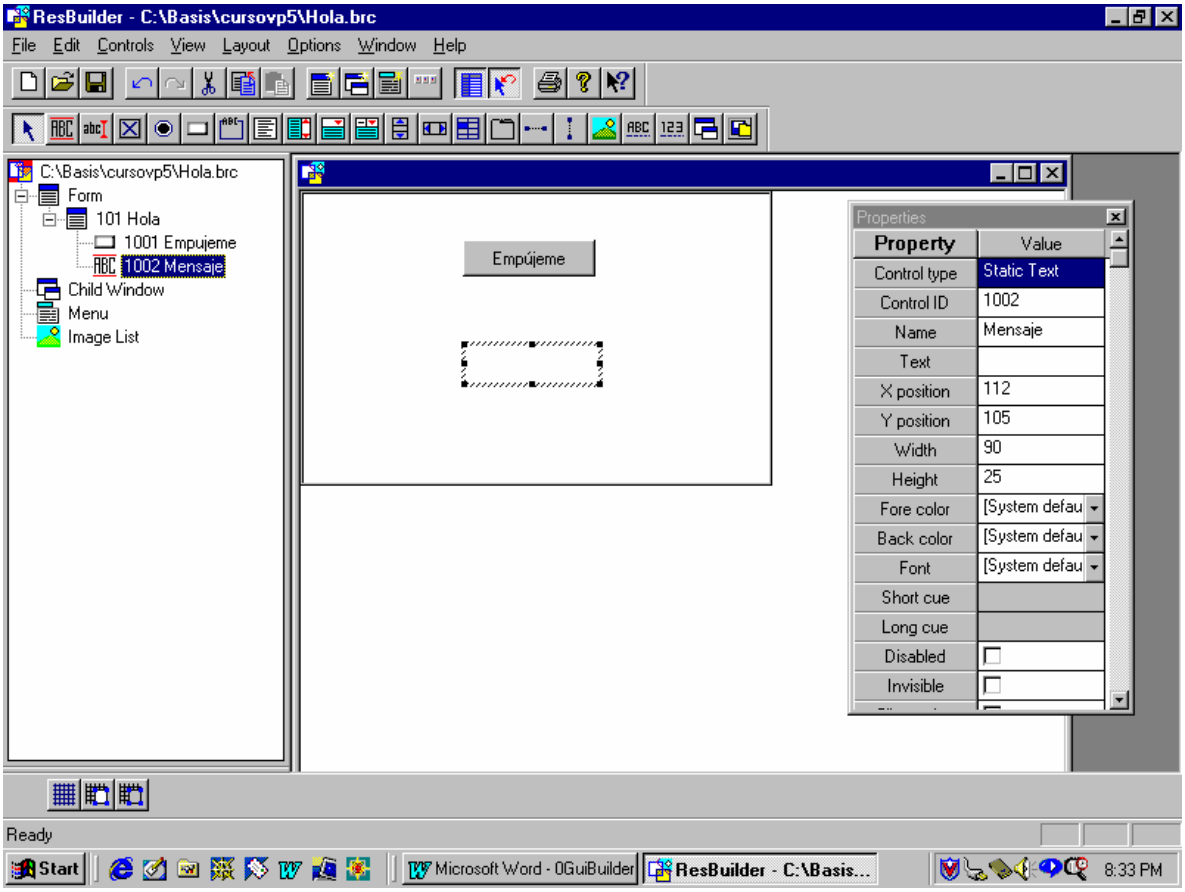
En un programa GUI, usted no tiene el mismo control directo sobre el flujo del programa como siempre usted lo ha tenido con un programa para ambiente de caracteres. En su lugar, usted crea una ventana con varios controles gráficos en esta y, en la esencia, se espera a que el usuario haga algo. Así a como un control es visible y habilitado, el usuario lo puede activar, por ejemplo, empujando un botón, escogiendo un ítem del menú, verificando una caja de chequeo, o haciendo clic en un botón de radio. Su programa necesita ser preparado para responder en una manera lógica a cualquier evento que el usuario cause o quiera hacer.

GUIBuilder™ hace un trabajo bueno al permitir controlar por completo los detalles técnicos del manejador de eventos del programa GUI, pero no puede ayudar con el diseño del programa. Porque un programa GUI opera muy diferentemente de un programa tradicional para ambiente de caracteres, y como recomendación, es una buena idea invertir tiempo alguna vez en leer y pensar acerca de temas relacionados al diseño con GUI.

Haciendo mi primer programa con GUIBuilder

Para iniciar siempre será necesario primero utilizar el ResBuilder, en este caso, para definir una muy simple forma con solo dos controles: un botón de pulsar y un control para texto estático. La forma se llamará **Hola**, el botón se denominará **Empújeme** y el control de texto se denominará **Mensaje**.

Su pantalla podría estarse viendo a como se muestra a continuación (no tiene que ser igual).



Hecho lo anterior hay que proceder a salvar la forma con el nombre **Hola.brc**.

Ahora si podrá levantar el GUIBuilder, activamos la opción del menú File e indicamos New. Esto hará que se nos pregunte el nombre para nuestro nuevo archivo GUIBuilder, al cual le vamos a poner **Hola**, que servirá de base para nuestro programa final. Hecho eso, GUIBuilder usará toda la información que pongamos en este archivo para generar y tokenizar un programa de trabajo de GUI, completo con su loop de eventos. Luego se nos preguntará si queremos correr ResBuilder para crear un recurso, pero como ya lo hicimos, hagamos clic en **No**. Entonces se nos pide que indiquemos el archivo de recursos que el programa usará. Entrar ahí el nombre del recurso **Hola.brc** que acabamos de crear.

Entonces un diálogo titulado **Program Options** aparece con una serie de parámetros a ser llenados. Dependiendo de lo que pongamos en esas opciones así trabajará el programa que luego sea generado. Por ejemplo, si queremos que nuestro programa tenga una declaración PREFIX especial, debemos entrar los directorios en ese primer campo. Dejar iguales los radio botones que ahí aparecen. Si quiere también puede hacer alguna referencia en el campo para derechos de autor.

Haciendo clic en el botón OK, retornaremos a la pantalla principal del GUIBuilder, y seleccionaremos la forma **Hola** desde la opción **Select Form** en el menú **Program**. La forma que acabamos de crear en ResBuilder aparecerá, con el botón **Empújeme** resaltado.

Entonces seleccionaremos un control con el que queramos trabajar. Uno fácil para iniciar es el objeto **close**, creado automáticamente por el ResBuilder y ubicado en la esquina superior derecha del contexto. Así que hagamos un doble clic sobre este, para que nos aparezca el recuadro pertinente a ese control, donde se redactan las instrucciones que serán ejecutadas cuando el usuario del programa haga clic sobre el objeto **close**.

Bien, aquí nos sale la pregunta: Qué es lo que el programa debería hacer cuando el usuario haga clic sobre el objeto **close**? Una acción equivalente a terminar el proceso dentro del programa GUI es finalizar el loop de eventos y tareas (end-of-job). Para finalizar el loop de eventos debe cambiar el valor de la variable que controla el loop por **`gb__eoj=1`**. Así es que agregue esa instrucción. (Sepa que los programas generados por GUIBuilder se saldrán del loop de eventos cuando todas las formas sean cerrados, sin que el programador explícitamente cambie la variable **gb__eoj**).

Ahora podemos continuar digitando las instrucciones que se ejecutarán cuando el usuario del programa haga clic sobre el botón **Empújeme**. Para eso es necesario volver a ver la forma. Una forma es dando clic en el botón select form (debajo del menú Tools). Entonces damos un doble click sobre el botón **Empújeme** y verá que se sobrepondrá una caja de la lista con tres entradas, muestrando los tres eventos que pueden ser accionados por ese botón. El primer evento posible es que el botón sea empujado. Las otras posibilidades son que el botón gane o pierda foco. De momento no nos interesa que el programa haga algo cuando el botón gane o pierda foco, pero si vamos a hacer que cuando el botón sea empujado, nos despliegue un mensaje en la pantalla, así que escojamos el primer evento.

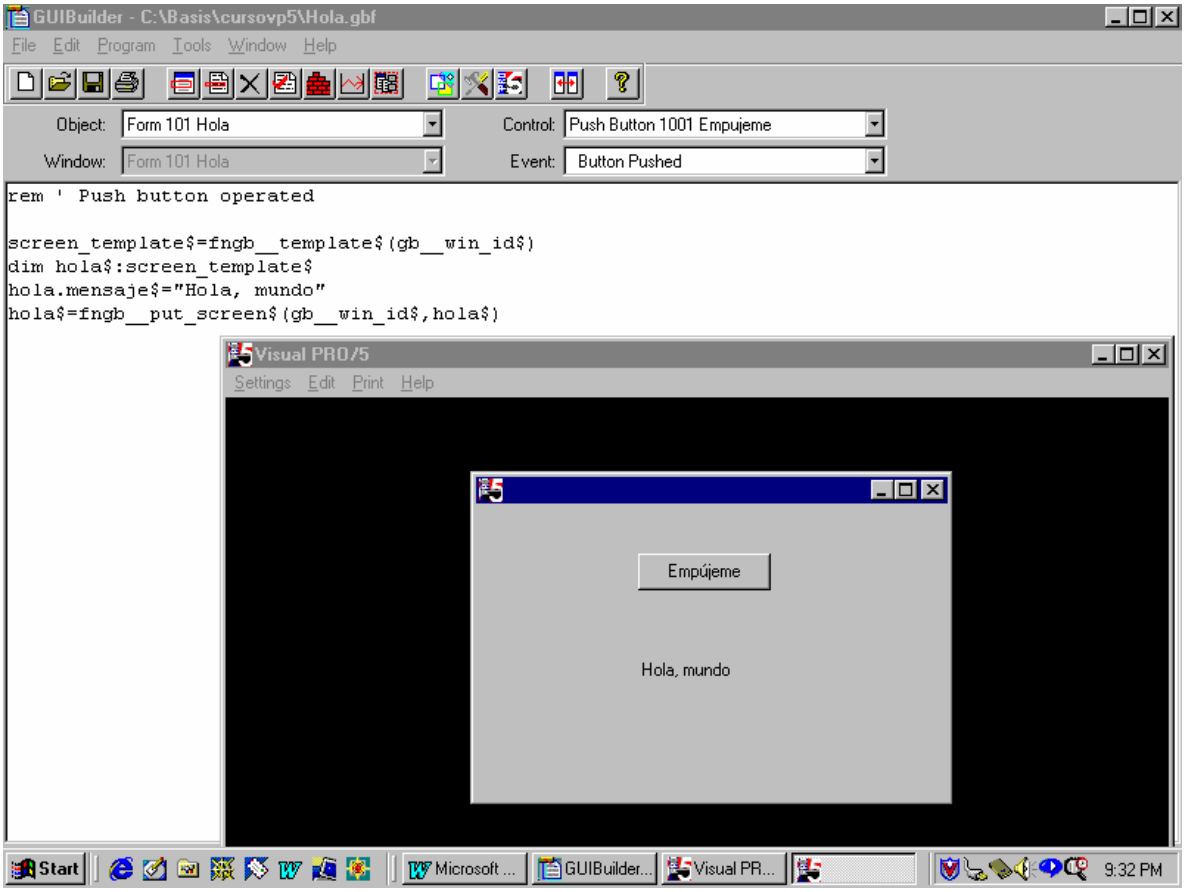
La pantalla principal de GUIBuilder nos traerá entonces un área en blanco para redacción de texto. No se alarme si nota que nuestra anterior declaración **gb__eoj=1** ya no aparece. Recuerde que GUIBuilder guarda las instrucciones en varios bloques de código, digamos, que esta es una forma mas moderna, que otros han encontrado como más ordenada y eficiente. Suponiendo que si realmente usted aún quiere programar al estilo viejo, entonces podría escribir subrutinas en GUIBuilder, compilando un programa pequeño, y escribiendo la finalización para que aparezca el >READY. De todos modos, continuemos y pongamos el código que se ejecutará cuando el usuario del programa haga clic sobre el botón **Empújeme**. Aunque de momento no le vea sentido a las siguientes cuatro líneas, por favor escríbalas:

```
screen_template$=fngb__template$(gb__win_id$)
```

```
dim hola$:screen_template$
hola.mensaje$="Hola, mundo"
hola$=fngb__put_screen$(gb__win_id$,hola$)
```

Para explicar lo que acaba de escribir, sepa que eso es una pareja de funciones proporcionadas al GUIBuilder para mostrar un mensaje en la pantalla. Existe la opción de usar un mnemónico de Visual® PRO/5 para lograr esto, pero las funciones proporcionan un código más legible, una vez que usted los aprendió. La primera línea invoca la función **`fngb__template$(),`** la cual nos retorna un TEMPLATE que nos describe todos los controles en la forma **Hola**. La segunda línea dimensiona la variable **Hola$** con ese TEMPLATE. Estamos escogiendo la variable **hola$** para que nos corresponda con el nombre que le asignamos en la forma en ResBuilder, pero no es un requerimiento que los nombres correspondan. En la tercera línea establecemos que la variable **hola.mensaje$** del TEMPLATE quede cargada con el valor que deseamos desplegar, el cual es "Hola, mundo". El nombre de variable **mensaje$** corresponde con el nombre del control para texto estático. Finalmente, en la cuarta línea se usa otra función para transferir los valores en el TEMPLATE a la pantalla.

Ahora ya estamos listos para compilar y correr nuestro programa. Seleccione **Run Program** y entre el nombre que quiera para el archivo del programa tokenizado. Ocurrirá que las cinco líneas de código que digitó , serán mezcladas y usadas para generar nuevas líneas de código, incluyendo la lógica del loop de eventos, y salvadas en un archivo ASCII fuente. Luego ese archivo fuente es compilado para obtener un así un programa para Visual PRO/5. GUIBuilder está haciendo correr así su programa **Hola Mundo**. Si hace un clic sobre el botón **Empújeme** se le aparecerá el mensaje en la pantalla.



Tendrá que admitir que obtener un programa GUI entero con sólo cinco líneas de código es algo impresionante. Si nunca ha tenido experiencia con TEMPLATES a la larga cuatro de las cinco líneas aún no sean de su entera comprensión. O tal vez esté pensando que si hubiera escrito el programa sin usar el GUIBuilder, le habría resultado más fácil, tal vez hasta usando números para IDentificar los controles de la forma. Pero cuando en la realidad usted llegue a trabajar con un contexto (forma) más complicado y lleno de controles, de fijo que verá la ventaja de usar una mejor herramienta como lo es el GUIBuilder.

Práctica con GUIBuilder**, para obtener un programa similar al que salvamos como "EJERCICIO1.pgm".**

1. Entrar al GUIBuilder.

2. Hacer Clic en el icono *New GUIBuilder file* para que se nos abra un contexto con nombre *Name New GUIBuilder file*. Ahí indicaremos EJERCICIO1 como nombre del programa que vamos a generar en el directorio Cursovp5.

3. Hecho lo anterior, se nos abre una ventana titulada *Create New Resource*, en la que se nos pregunta si necesitamos crear un nuevo archivo de Recursos, a lo que responderemos que No. Con eso se nos abre otra ventana llamada *Open Resource File*, para que demos OPEN al archivo **Ejercicio1.brc**. De seguido se nos abre la ventana llamada *Program Options*, a la cual le puede dar OK así a como le aparece.

4. De aquí en adelante ya se nos muestra la pantalla principal del GUIBuilder, mediante la cual podemos digitar el código que nos permitirá controlar los diferentes eventos que le vayan a ocurrir al programa, así como las diferentes rutinas necesarias para su funcionamiento. Para eso tenemos dos caminos: uno es seleccionando la forma que vamos a trabajar (5to icono) y el otro es seleccionando los Objetos, Ventanas, Controles y Eventos que aparecen en las cajas de listas que ahí se muestran.

5. Hagamos Clic en el 5to icono de la barra de herramientas (Select Form) y escojamos la línea que dice *101 frm_Ejercicio*. Eso hace que nos aparezca una pantalla que nos muestra en forma desabilitada la primera de las dos pantallas que hicimos con el ResBuilder.

6. De aquí en adelante realmente se tiene la libertad de empezar a digitar el código para los diferentes objetos y/o rutinas en el orden en que se quiera hacer. Uno de estos eventos puede ser el de terminar el proceso por medio de la X en la esquina superior derecha. Para eso hagamos un Clic sobre dicha X y en el contexto que se nos abre para el evento *Window Closed* agreguemos el siguiente código:

   ```
   gb__eoj=1
   ```

   Lo anterior causará que cuando el usuario del programa haga Clic sobre dicha X, la variable de GUIBuilder **gb__eoj** cambie su valor a un uno, lo cual causará que se active la rutina de eventos predeterminada por GUIBuilder como * *End Of Job Code*. A esta rutina podemos entrarle el código necesario para dar la adecuada terminación al proceso del programa. Pude ser el retorno a un menú o el retorno del Visual PRO/5 hacia Windows. Para lograr eso abrimos el control de lista *Object* y seleccionamos la rutina * *End Of Job Code* para que digitemos por ejemplo:

   ```
   rem ' ------------------------------------------------------
   rem ' EOJ
   rem ' ------------------------------------------------------
   ```

```
RELEASE
```

7. El siguiente paso a codificar podría ser aquel en el que abrimos los archivos o base de datos a ser usados. Ahí también definimos las variables y constantes necesarias para el proceso posterior del programa. Esto tenemos que hacerlo en una rutina también predeterminada por el GUIBuilder como * *Initialization Code*. Igual a como lo hicimos en el paso 6, la buscamos y digitamos lo siguiente:

```
rem ' ------------------------------------------------------
rem ' Init
rem ' ------------------------------------------------------

Print (gb__sysgui)'context'(gb__win.frm_Ejercicio)

rem ' ------------------------------------------------------
rem ' DEFINE CONSTANTES
REM ' para obtener informacion de campos usados en el contexto, por
REM ' ejemplo: TXT_CLIENTE es el name asignado en el ResBuilder,
REM ' y por medio de este nombre obtenemos el numero de id que se le
REM ' asignó en el ResBuilder, que es mejor guardar en una variable.
REM '------------------------------------------------------
DIM FRM_SCREEN$:FNGB__TEMPLATE$(GB__WIN_ID$)
   ID_CLIENTE=NUM(FATTR(FRM_SCREEN$,"TXT_CLIENTE","ID"))
   ID_NOMBRE=NUM(FATTR(FRM_SCREEN$,"LST_NOMBRE","ID"))
   ID_DIRECCION=NUM(FATTR(FRM_SCREEN$,"TXT_DIRECCION","ID"))
   ID_TELEFONO=NUM(FATTR(FRM_SCREEN$,"TXT_TELEFONO","ID"))
   ID_FECHA_ING=NUM(FATTR(FRM_SCREEN$,"TXT_FECHA_ING","ID"))
   ID_LIMITE=NUM(FATTR(FRM_SCREEN$,"TXT_LIMITE","ID"))
   ID_BUSCAR=NUM(FATTR(FRM_SCREEN$,"BTN_BUSCAR","ID"))
   OBTENER_TEXTO=1
REM ' ------------------------------------------------------

REM 'Abre base de datos a ser usada

SQLCHAN=SQLUNT
SQLOPEN(SQLCHAN)"Cuentas por Cobrar"

GOSUB LISTA_NOMBRES

SQLPREP(SQLCHAN)"SELECT * FROM CLIENTES"
DIM CLIEN$:SQLTMPL(SQLCHAN)
PRINT (GB__SYSGUI)'FOCUS'(ID_CLIENTE)
```

8. La rutina `LISTA_NOMBRES` mencionada en la anterior rutina es la que nos carga en memoria la lista con los nombres de todos los clientes. Si queremos podemos escribirla de una vez, para lo cual abrimos el control de lista *Object* y seleccionamos --- *New Subroutine/Function* ---. Nos aparece una ventana *Name Subroutine/Function* para que digitemos:

```
LISTA NOMBRES
```

Le damos aceptar y procedemos a digitar el siguiente código:

```
rem ' ------------------------------------------------------
rem ' LISTA NOMBRES
rem ' ------------------------------------------------------

LISTA_NOMBRES:

SQLPREP(SQLCHAN)"SELECT CODIGO,NOMBRE FROM CLIENTES
: ORDER BY NOMBRE"
SQLEXEC(SQLCHAN)
DIM CLI$:SQLTMPL(SQLCHAN)
PRINT(GB__SYSGUI)'LISTCLR'(ID_NOMBRE)
LEE_CLIENTES:
CLI$=SQLFETCH(SQLCHAN,ERR=FIN_LISTA)
PRINT(GB__SYSGUI)'LISTADD'(ID_NOMBRE,-1,CLI.NOMBRE$+" "+CLI.CODIGO$)
GOTO LEE_CLIENTES
FIN_LISTA:

return
```

9. Para continuar, podemos entrar el código necesario para los campos o controles donde el usuario del programa luego tendrá que digitar información. Repetir entonces el paso 5.
   Damos un doble Clic sobre el control donde va a ser digitado el código del cliente (INPUTE **200 txt_cliente**), seleccionamos el evento **** Control Lost Focus***, para agregar luego el siguiente código:

```
CLIEN.CODIGO$=CTRL(GB__SYSGUI,ID_CLIENTE,OBTENER_TEXTO)
IF CVS(CLIEN.CODIGO$,2)="" THEN RETURN
GOSUB VERIFICA_CODIGO
```

10. Como acabamos de mencionar el nombre de la rutina `VERIFICA_CODIGO`, podemos proceder a escribirla de una vez. Para eso abrimos el control de lista *Object* y seleccionamos *--- New Subroutine/Function ---*. Nos aparece una ventana *Name Subroutine/Function* en donde digitamos:

```
VERIFICA CODIGO
```

Damos aceptar y escribimos el siguiente código:

```
rem ' ------------------------------------------------------
rem ' VERIFICA CODIGO
rem ' ------------------------------------------------------

VERIFICA_CODIGO:

  SQLPREP(SQLCHAN)"SELECT * FROM CLIENTES
:   WHERE CODIGO='"+CLIEN.CODIGO$+"'"
  SQLEXEC(SQLCHAN)
```

```
  DIM CLI$:SQLTMPL(SQLCHAN)
  CLI$=SQLFETCH(SQLCHAN,ERR=NUEVO_REGISTRO)
  GOTO DESPLEGAR_REGISTRO

NUEVO_REGISTRO:
  GOSUB LIMPIAR_PANTALLA

return
```

11. Dentro de la subrutina anterior estamos mencionando otra subrutina llamada LIMPIAR_PANTALLA. La podemos hacer de una vez y observar que a excepción del campo para la dirección, todos los campos pueden ser limpiados en una misma instrucción. Entonces de nuevo abrimos el control de lista *Object*, seleccionamos --- *New Subroutine/Function* --- y cuando nos aparece la ventana *Name Subroutine/Function* digitamos:

```
LIMPIAR PANTALLA
```

Damos aceptar y escribimos:

```
rem ' ------------------------------------------------------------
rem ' LIMPIAR PANTALLA
rem ' ------------------------------------------------------------

LIMPIAR_PANTALLA:

Print (GB__SYSGUI)'CLRTITLE'(ID_NOMBRE,ID_TELEFONO,ID_FECHA_ING,ID_LIMITE)
Print (GB__SYSGUI)'TXCLR'(ID_DIRECCION)

Return
```

12. Dentro del código digitado en el paso 10 también mencionamos otra rutina llamada DESPLEGAR_REGISTRO, la cual cumple con la función de pasar los datos del registro a los controles de la forma. Abramos el control de lista *Object*, seleccionemos --- *New Subroutine/Function* ---, entramos a la ventana *Name Subroutine/Function* y digitamos:

```
DESPLEGAR REGISTRO
```

Damos aceptar y digitamos:

```
rem ' ------------------------------------------------------------
rem ' DESPLEGAR REGISTRO
rem ' ------------------------------------------------------------

DESPLEGAR_REGISTRO:

  GOSUB LIMPIAR_PANTALLA
  PRINT(GB__SYSGUI)'TITLE'(ID_CLIENTE,CLI.CODIGO$)
  PRINT(GB__SYSGUI)'TITLE'(ID_NOMBRE,CLI.NOMBRE$)
  PRINT(GB__SYSGUI)'TXADD'(ID_DIRECCION,-1,CLI.DIRECCION$)
  PRINT(GB__SYSGUI)'TITLE'(ID_TELEFONO,CLI.TELEFONO$)
  PRINT(GB__SYSGUI)'TITLE'(ID_FECHA_ING,DATE(CLI.FECHA_ING:"%Dz%Mz%Yl"))
```

```
    PRINT(GB__SYSGUI)'TITLE'(ID_LIMITE,STR(CLI.LIMITE_CR))

  return
```

13. El siguiente campo que podemos controlar es el objeto donde se digitará o seleccionará el nombre del cliente (List Edit **201 lst_Nombre**). Repetimos el paso 5, damos un doble Clic sobre ese control y seleccionamos el evento **** List Selection***, para agregar luego el siguiente código:

Se puede decir que hasta aquí hemos entrado el código necesario para controlar los eventos sobre los campos en donde luego va a ser puesta la información de los clientes. Sobre este contexto se nos está quedando por programar los eventos para los botones colocados en la parte superior izquierda de la pantalla, así como también

```
rem ' Notify Event - List Edit Control - List Selection
NOMBRE$=CTRL(GB__SYSGUI,ID_NOMBRE,OBTENER_TEXTO)
if CVS(NOMBRE$,2)="" then RETURN
CLIEN.CODIGO$=NOMBRE$(LEN(CLIEN.NOMBRE$)+2)
GOSUB VERIFICA_CODIGO
```

14. nos falta el código para las diferentes opciones de menú que indicamos desde el ResBuilder.

15. Variemos un poco la forma de elegir los objetos a programar y hagamos Clic sobre la caja de lista llamada *Control:*, seleccionemos el evento *Menu Item* ***11 Nuevo*** y agreguemos el siguiente código:

```
PRINT(GB__SYSGUI)'CLRTITLE'(ID_CLIENTE)
GOSUB LIMPIAR_PANTALLA
PRINT (GB__SYSGUI)'FOCUS'(ID_CLIENTE)
```

16. Siempre sobre la caja de lista llamada *Control:*, escojamos el evento *Menu Item* ***12 Grabar*** y agreguemos:

```
GOTO GRABAR
```

17. Ahí mismo seleccionemos ahora *Menu Item* ***13 Eliminar*** y agreguemos:

```
GOTO ELIMINAR
```

18. Seguimos con *Menu Item* ***15 Imprimir***, en donde agregamos:

```
GOTO IMPRIMIR
```

19. Agregar en *Menu Item* ***17 Salir***:

```
gb__eoj=1
```

20. Ahora sigamos con los Tool Button, ya sea haciendo Clic sobre estos o seleccionándolos directamente desde la caja de lista llamada *Control:*. Escojamos el evento *Tool Button* ***300 btn_Nuevo*** y agreguemos:

```
PRINT(GB__SYSGUI)'CLRTITLE'(ID_CLIENTE)
GOSUB LIMPIAR_PANTALLA
PRINT (GB__SYSGUI)'FOCUS'(ID_CLIENTE)
```

21. Sigamos con el *Tool Button* ***301 btn_Guardar*** y agreguemos:

```
GOTO GRABAR
```

22. Al *Tool Button* ***302 btn_Eliminar*** le agregamos:

```
GOTO ELIMINAR
```

23. Al *Tool Button* ***303 btn_Imprimir*** le agregamos:

```
GOTO IMPRIMIR
```

24. Al *Tool Button* ***304 btn_Salir*** le agregamos:

```
gb__eoj=1
```

25. Al *Tool Button* ***305 btn_Buscar*** le agregamos:

```
GOSUB BUSQUEDA
```

26. Para esas nuevas subrutinas que se invocan desde esos Tool Button o desde las opciones de menú (Grabar, Eliminar, Imprimir y Busqueda), debemos proceder a entrar su código. Hagamos primero la de GRABAR, para lo cual abrimos el control de lista *Object*, seleccionamos --- *New Subroutine/Function* ---, entramos a la ventana *Name Subroutine/Function* y digitamos:

```
GRABAR
```

Damos aceptar y digitamos:

```
rem ' ----------------------------------------------------------------
rem ' GRABAR
rem ' ----------------------------------------------------------------

GRABAR:
  CLIEN.CODIGO$=CTRL(GB__SYSGUI,ID_CLIENTE,OBTENER_TEXTO)
  if CVS(CLIEN.CODIGO$,2)="" then
:       M=msgbox("No ha digitado el código de cliente",64,"Código");return
  clien.nombre$=ctrl(GB__SYSGUI,ID_NOMBRE,OBTENER_TEXTO)
  if CVS(CLIEN.NOMBRE$,2)="" then
:       M=msgbox("No ha digitado el nombre",64,"Nombre");RETURN
  clien.direccion$=CTRL(GB__sysgui,id_direccion,obtener_texto)
  clien.telefono$=CTRL(GB__sysgui,id_telefono,obtener_texto)

  fecha$=CTRL(GB__sysgui,id_fecha_ing,obtener_texto)
```


Manual preparado en Costa Rica por Daniel Ortíz de SPS Software <docsps@racsa.co.cr>
y Jorge Garro de UniSoft <unisoft@racsa.co.cr>

```
  clien.fecha_ing=JUL(NUM(fecha$(5,4)),NUM(fecha$(3,2)),NUM(fecha$(1,2)),
: ERR=fecha_incorrecta)

  clien.limite_cr=NUM(CTRL(GB__sysgui,id_limite,obtener_texto))

  SQLPREP (sqlchan)"Insert into Clientes values(?,?,?,?,?,?)"
  SQLEXEC (sqlchan,ERR=actualizar)clien.codigo$,clien.nombre$,
:        clien.direccion$,clien.telefono,clien.fecha_ing,clien.limite_cr

  GOSUB LISTA_NOMBRES
  GOSUB LIMPIAR_PANTALLA
return

ACTUALIZAR:
  M=msgbox("El registro ya existe, desea Actualizarlo",4+32,"Ya Existe")
  if M=7 then RETURN
  SQLPREP (sqlchan)"Update Clientes set Nombre=?,Direccion=?,TELEFONO=?,
:   FECHA_ING=?,LIMITE_CR=? WHERE CODIGO='"+clien.codigo$+"'"
  SQLEXEC (sqlchan)clien.nombre$,clien.direccion$,clien.telefono,
:   clien.fecha_ing,clien.limite_cr

  GOSUB LISTA_NOMBRES
  GOSUB LIMPIAR_PANTALLA

return
```

27. Sigamos con la subrutina de ELIMINAR, abrimos el control de lista *Object*, seleccionamos --- *New Subroutine/Function* ---, entramos a la ventana *Name Subroutine/Function* y digitamos:

    ELIMINAR

    Damos aceptar y digitamos:

```
rem ' ---------------------------------------------------------------
rem ' ELIMINAR
rem ' ---------------------------------------------------------------

ELIMINAR:
  CLIEN.CODIGO$=CTRL(GB__SYSGUI,ID_CLIENTE,OBTENER_TEXTO)
  if CVS(CLIEN.CODIGO$,2)="" then RETURN
  SQLPREP(SQLCHAN)"SELECT * FROM CLIENTES
:        WHERE CODIGO='"+CLIEN.CODIGO$+"'"
  SQLEXEC(SQLCHAN)
  DIM CLI$:SQLTMPL(SQLCHAN,ERR=NO_EXISTE_REGISTRO)
  M=msgbox("Desea eliminar el cliente"+clien.codigo$,4+32,"Eliminar")
  if M=7 then RETURN
  SQLPREP (sqlchan)"DELETE FROM CLIENTES WHERE CODIGO='"+clien.codigo$+"'"
  SQLEXEC (sqlchan)
  GOSUB LIMPIAR_PANTALLA
  Print (GB__SYSGUI)'CLRTITLE'(ID_CLIENTE)
  GOSUB LISTA_NOMBRES
  Print (GB__SYSGUI)'FOCUS'(ID_CLIENTE)
```

```
NO_EXISTE_REGISTRO:

return
```

28. Siguiendo con el orden arriba indicado, nos corresponde escribir el código para la subrutina IMPRIMIR, para lo cual abrimos el control de lista *Object*, seleccionamos --- *New Subroutine/Function* ---, entramos a la ventana *Name Subroutine/Function* y digitamos:

```
IMPRIMIR
```

Damos aceptar y digitamos:

```
rem ' ------------------------------------------------------------------
rem ' IMPRIMIR
rem ' ------------------------------------------------------------------

IMPRIMIR:
  PCHAN=UNT
  OPEN(PCHAN,MODE="PREVIEW,COLS=132",ERR=IMPRESORA_OCUPADA)"LP"

  L=100,L1=54,P=0,TOTAL_CLIENTES=0
  SQLPREP(SQLCHAN)"SELECT * FROM CLIENTES"
  SQLEXEC(SQLCHAN)
  DIM CLI$:SQLTMPL(SQLCHAN)

REPORTE_CLIENTES:
  CLI$=SQLFETCH(SQLCHAN,ERR=FIN_DE_REPORTE)
  if L>L1 then GOSUB ENCABEZADO
  Print (PCHAN)CLI.CODIGO$," ",CLI.NOMBRE$," ",CLI.TELEFONO$
  L=L+1, TOTAL_CLIENTES=TOTAL_CLIENTES+1
  GOTO REPORTE_CLIENTES

FIN_DE_REPORTE:
  if TOTAL_CLIENTES=0 then GOTO CERRAR_IMPRESORA
  if L+3 > L1 then GOSUB ENCABEZADO
  Print (PCHAN)FILL(132,"-")
  Print (PCHAN)"TOTAL DE CLIENTES: ",TOTAL_CLIENTES
  Print (PCHAN)FILL(132,"-")

CERRAR_IMPRESORA:
CLOSE(PCHAN)

Return

rem ' ------------------------------------------------------------------
rem ' ENCABEZADO
rem ' ------------------------------------------------------------------

ENCABEZADO:

  L=0,P=P+1
  if P>1 then Print (PCHAN)'FF',
  Print (PCHAN)"CURSO DE VISUAL",@(75),"Pagina:",P,'LF',"Sistema de
: cuentas por cobrar",'lf',"Reporte de clientes",'LF',fill(80,"-"),'LF',
```

```
: "Cliente Nombre",@(30),"Telefono",'LF',fill(80,"-")

return
```

29. Sigamos con el código de la rutina para BUSQUEDA, para lo cual abrimos el control de lista *Object*, seleccionamos --- *New Subroutine/Function* ---, entramos a la ventana *Name Subroutine/Function* y digitamos:

```
BUSQUEDA
```

Damos aceptar y digitamos:

```
rem ' ------------------------------------------------------------
rem ' BUSQUEDA
rem ' ------------------------------------------------------------

BUSQUEDA:
  rem ' para posicionarme en la otra ventana
  REM ' ------------------------------------------------------------
  Print (GB__SYSGUI)'CONTEXT'(GB__WIN.Frm_Consulta)

  rem ' para hacerla visible, ya que desde un inicio no es visible
  REM ' ------------------------------------------------------------
  Print (GB__SYSGUI)'SHOW'(0)
  Print (GB__SYSGUI)'FOCUS'(101)

  Return
```

30. Puesta en ejecución la anterior subrutina, le aparecerá al usuario el segundo contexto que habíamos dibujando con el ResBuilder (102 Consulta), para el cual también debemos digitar código como por ejemplo, el que se requiere para cuando hagan Clic sobre la cajita de cierre ubicada en la esquina superior derecha del contexto. Para lograr eso hagamos Clic en el 5to icono de la barra de herramientas (Select Form) y escojamos la línea que dice *102 Consulta*. Eso hace que nos aparezca la segunda pantalla en forma desabilitada. Para hacerlo diferente de como lo hicimos con la primer pantalla, demos un doble Clic sobre la superficie de esa pantalla (que no sea sobre alguno de sus objetos), para que nos salga una ventana llamada *Select Event* que nos muestra los diferentes eventos que podrían darse para la forma 102. Entonces buscamos que el que se llama * *Window Closed* y digitamos el siguiente código:

```
rem ' Close Button Operated

REM '  PARA SALIR DEL CONTEXTO O VENTANA DE BUSQUEDA
REM ' ------------------------------------------------------------

Print (GB__SYSGUI)'CONTEXT'(GB__WIN.Frm_Consulta)
Print (GB__SYSGUI)'HIDE'(0)
Print (GB__SYSGUI)'CONTEXT'(GB__WIN.frm_Ejercicio)
```

Lo anterior causará que la pantalla o contexto para consultas desaparezca y nos vuelva a quedar visible la pantalla principal.

31. Para continuar, podemos digitar el código requerido para la cajita donde vamos a capturar el nombre de cliente a ser buscado. Hagamos Clic en la caja de lista *Control* y seleccionemos *INPUTE 101 InputE*, y luego en la caja de lista *Event* seleccionemos * *Control Lost Focus* para entrar estas instrucciones:

```
rem ' Control lost focus
Print (GB__SYSGUI)'CONTEXT'(GB__WIN.Frm_Consulta)

NOMBRE$=CTRL(GB__SYSGUI,101,OBTENER_TEXTO)
if CVS(NOMBRE$,2)="" then RETURN

SQLPREP(SQLCHAN)"SELECT CODIGO,NOMBRE FROM CLIENTES
: WHERE NOMBRE >='"+NOMBRE$+"' ORDER BY NOMBRE"
SQLEXEC(SQLCHAN)
DIM CLI$:SQLTMPL(SQLCHAN)
PRINT(GB__SYSGUI)'LISTCLR'(102)
LEE_NOMBRES:
CLI$=SQLFETCH(SQLCHAN,ERR=FIN_NOMBRES)

if cli.nombre$(1,len(nombre$))>nombre$ then goto fin_nombres

PRINT (GB__SYSGUI)'LISTADD'(102,-1,CLI.NOMBRE$+" "+CLI.CODIGO$)
GOTO LEE_NOMBRES
FIN_NOMBRES:
```

Lo anterior equivale a forzar una lectura de registros, ordenada por nombre de cliente, para que nos sean mostrados todos los que sean inicialmente iguales al dato almacenado en la variable NOMBRE$.

32. El código anterior nos cargaría en memoria la lista de nombres requeridos para seleccionar el que interese al usuario. Para esto ahora debemos entrar el código necesario para *el Control: List Box **102 List Box***, el cual debemos seleccionar arriba junto al *Event: *** Clicked on List***.

```
rem ' Click in list box

Print (GB__SYSGUI)'CONTEXT'(GB__WIN.Frm_Consulta)

NOMBRE$=CTRL(GB__SYSGUI,102,OBTENER_TEXTO)
if CVS(NOMBRE$,2)="" then RETURN

CLIEN.CODIGO$=NOMBRE$(LEN(CLIEN.NOMBRE$)+2)

Print (GB__SYSGUI)'HIDE'(0)

Print (GB__SYSGUI)'CONTEXT'(GB__WIN.frm_Ejercicio)
GOSUB VERIFICA_CODIGO
```

33. Ya casi terminamos. Solo nos falta suministrar el código necesario para que una serie de eventos que puedan ocurrir, sean notificados al usuario del programa. Hagamos clic en la cajita *Object* y digitamos una nueva subrutina llamada **Mensajes**

```
rem ' ------------------------------------------------------------
rem ' MENSAJES
rem ' ------------------------------------------------------------

MENSAJES:

fecha_incorrecta:
LET m=MSGBOX("La fecha es incorrecta, debe utilizar el formato
DD/MM/AAAA
: para "+$0A$+"el registro de fechas en el programa",16,"Error Fecha")
PRINT (GB_SYSGUI)'FOCUS'(id_fecha)
return

REM ----------------------------------
falta_codigo:
LET m=MSGBOX("No ha digitado el codigo de cliente",16,"Codigo")
PRINT (GB_SYSGUI)'FOCUS'(id_cliente)
return

REM ----------------------------------
falta_nombre:
LET m=MSGBOX("No ha digitado el nombre del cliente",16,"Nombre")
PRINT (GB_SYSGUI)'FOCUS'(id_nombre)
return

REM ----------------------------------
impresora_ocupada:
LET m=MSGBOX("La impresora esta ocupada",16,"Error en Impresora")
return
```

Con lo anterior, podemos decir que el programa está terminado y podemos proceder a probarlo. Entre clientes nuevos, luego hágales cambios, elimine registros y pruebe los dos tipos de búsqueda, de los cuales se puede decir que el segundo es el que se recomienda para casos en la cantidad de registros sea masiva, ya que con la primer consulta el programa constantemente está recreando la lista de nombres y eso equivale a leer o barrer todos los registros en la tabla de clientes.

Sepa que el programa este perfectamente le puede servir de machote para crear otros similares.

Programa de Mantenimiento utilizando el Control Grid y GuiBuilder

En esta nueva práctica repasaremos un poco lo ya aprendido, así como también haremos la introducción al Grid, el cual es un tipo de objeto cuadriculado compuesto de líneas y columnas, muy similar a una hoja electrónica.

Existen dos tipos de Grid
- El que se 'pega' a un archivo en donde cada línea del objeto contendrá los datos de cada uno de los registros del archivo y donde cada columna contiene el dato de cada uno de los campos de los registros. Las columnas son mostradas exactamente en el mismo orden en que están definidos en el diccionario de datos.
- El otro tipo de Grid es el que se trabaja de una manera libre en donde no se tiene un ligamiento directo hacia un archivo. Un ejemplo de su uso es por ejemplo, en un programa de facturación u ordenes de compra, en el que el usuario tiene que ir digitando los códigos y cantidades y el programa le va poniendo las descripciones y precios de los productos.

Lo que haremos en esta práctica es un programa de mantenimiento que manejará un Grid 'pegado' a un archivo Maestro de Empleados, para lo cual necesitaremos:

- Definir una nueva tabla de datos en el diccionario de datos (con el DDBuilder)
- Crear el correspondiente Contexto de pantalla (con el ResBuilder)
- Usando GuiBuilder, escribir el código necesario para que el programa funcione adecuadamente.

Definición de la tabla de datos

Ingresar entonces al DDBuilder para hacer la definición de una nueva estructura de datos:

1) En la opción 'File' del menú hacemos clic en 'Open config' y procedemos a buscar el mismo archivo **.tpm** que estamos usando en el curso: **c:\basis\cursovp5\curso.tpm**

2) Luego haciendo un clic con el botón derecho del ratón sobre la opción **Tables** escojemos la opción **Add** y donde aparece el nombre Table1 que inicialmente asume el DDBuilder, ponemos el nombre **Maest_pl**.

3) Seguidamente llenamos los siguientes campos:
   - Description: Maestro de empleados
   - Path: (DATA)Maest_pl

4) Luego hacemos clic derecho sobre **Columns** para proceder a incluir los siguientes campos:

   - CODIGO C(6)
   - NOMBRE C(30)
   - CEDULA C(20)

Manual preparado en Costa Rica por Daniel Ortíz de SPS Software <docsps@racsa.co.cr>
y Jorge Garro de UniSoft <unisoft@racsa.co.cr>

- DIRECCION C(50)
- TELEFONO C(10)
- FECHA_INGRESO N(8)
- TIPO_PLANILLA C(1)
- SALARIO N(10)
- SEXO N(1)

5) Hecho lo anterior datos un doble clic sobre **Indices** y luego un clic derecho, para definir como única llave de acceso el campo CODIGO.

6) Luego, hacemos un clic sobre la línea en donde se muestra el nombre Maest_pl, para que luego hagamos clic sobre el icono **Make** (de la barra de herramientas), para obtener así la definición de la estructura en el disco.

7) Terminamos haciendo clic en el icono que muestra un disket, para hacer el **Save** a todas las definiciones que hemos dado.

Creación del Contexto (Pantalla)

Accese el Resbuilder y seleccione nuevo Resource File desde el menú File para crear un nuevo archivo de recurso.
Seleccione Add Form desde el menú Edit para crear un form numerado con el ID 101 en la ventana de propiedades cambie los siguientes valores:

Title	**Mantenimiento al Maestro de Empleados**
Name	frmgrid
Current Units:	**semichars**
X position	20
Y position	60
Width	365
Height	200

Adicione a la forma un Grid Control seleccionándolo de la barra de controles y colóquelo en cualquier parte del form; en la ventana de propiedades coloque los siguientes valores

ID	204
Name	grdmaestroGrid
X position	8
Y position	42
Width	348
Height	138
Num Rows	1
Num Columns	9

Row Head Width	15 Nota: (Habilitar botón Row Head y después deschequearlo)
Col Head	checked
Horiz Scroll	checked
Vert Scroll	checked

En Propiedades de Columna (Column Prop) seleccione el Botón (...)
Deje el título de la columna en blanco y coloque los siguientes valores

Column 1 Width	45
Column 2 Width	100
Column 3 Width	50
Column 4 Width	100
Column 5 Width	40
Column 6 Width	40
Column 7 Width	10
Column 8 Width	50
Column 9 Width	10

Ahora en la parte superior del contexto adicione cuatro botones Push Button y coloque las siguientes propiedades para cada uno

Primer botón	
ID	200
Name	btnNuevo
Text	Nuevo
X position	7
Y position	6
Width	25
Height	15

Segundo Botón	
ID	201
Name	btnEditar
Text	Editar
X position	32
Y position	6
Width	25
Height	15

Tercer Botón	
ID	202
Name	btnBorrar
Text	Borrar
X position	57
Y position	6
Width	25

Height	15

Cuarto Botón

ID	203
Name	btnSalir
Text	Salir
X position	82
Y position	6
Width	25
Height	15

Salve el archivo de recurso en el directorio \Basis\cursovp5\ como **plani3.brc**

Creación del Programa

Accese el GuiBuild y seleccione New en el menú File y cree un nuevo Guibuilder File, en la caja de diálogo escriba **plani3** como nombre para el nuevo programa. Cuando aparezca el cuadro de diálogo Create new Resource File de Clic en **NO,** seleccione el archivo de pantalla **plani3.brc** y de open, de Ok en la ventana de propiedades del programa para aceptar los valores por default.
Ahora el recurso ya queda ligado al programa y el desarrollo del código para el Grid puede comenzar

Primero seleccione **End of Job Code** desde el Botón de Lista Object, un encabezado de End of Job aparecerá en el área de edición del Guibuilder, escriba **release** en una línea nueva después de las marcas, el código de End of Job es ejecutado cuando el programa termina

Para hacer que el botón de salir sea funcional seleccione Form 101 frmGrid desde el Botón de Lista de Object y en el Botón de Lista de Control seleccione Push Button 203 btnSalir. En el Botón de Lista de eventos seleccione Button Pushed, en el área de edición del Guibuilder digite después del encabezado una nueva línea con lo siguiente

```
gb__eoj=1
```

este código será ejecutado cuando el usuario de click en el botón de Salir mientras el programa esté corriendo. Esto pone una bandera de verdadero y causa que el loop de eventos termine y ejecute el End of Job.
Note que existen variables y nombres de funciones especiales en el Guibuilder indicadas siempre con **gb** seguido por dos underscores (**gb__**).

Salvemos el programa haciendo clic sobre el disket que se muestra en la barra de herramientas y luego, aunque no esté terminado podemos ponerlo en ejecución, para lo cual tenemos dos opciones. Si quiere haga clic en el menú **Program** y seleccione ahí el ícono **Run Program**. El programa entonces correrá y desplegará la pantalla creada. De click en el botón Salir, el programa terminará.
Entre de nuevo al Guibuilder y abra el archivo **plani3.gbf**

Haciendo que el GRID funcione

Primero escriba el código de inicialización, empiece seleccionando **Initialization Code** desde el Botón de Lista **Object,** un encabezado de código de inicialización aparecerá en el área de edición del Guibuilder, adicione el siguiente código:

```
rem obtener un template que describe la forma que se usa
dim datagrid_temp$:fngb__template$(gb__win_id$)

rem define constantes
gosub Define_constantes

rem abre el archivo de datos y obtiene el template
gosub Abrir_archivo

rem hacer que el grid trabaje con datos
gosub Atar_grid_al_canal
```

Ahora, para crear las tres diferentes subrutinas siga los siguientes pasos:

Seleccione **New Subroutine/Function** en el Botón de Lista de **Object** y en el cuadro de diálogo digite **Define Constantes** y Ok, esto colocará un encabezado en el área de edición del Guibuilder, digite en una nueva línea las siguientes instrucciones. (Si quiere evitarse esta digitación puede optar por copiarla con el mouse desde el tutorial en línea que se ofrece para el Grid. Vaya a la documentación en línea que trae el Visual PRO/5 y haga un 'Find' con **Data-Aware Grid Tutorial**. Las instrucciones a ser copiadas están en la sección 5, y sepa que son iguales para todo Grid atado a un archivo).

```
Define_constantes:

rem message box constants
msgboxYes=6
msgboxYesNo=4
msgboxExclamation=48
msgboxInfo=64
msgboxSecond=256

rem grid send message functions
gridSetHeadingTitles=23
gridEndEdit=26
gridStartEdit=31
gridGetEdit=34
gridSetEdit=35
gridGetNumberofCols=40
gridGetNumberofRows=41
gridGetSelectedCol=44
```

```
gridGetSelectedRow=45
gridGotoCol=47
gridGotoRow=48
gridShowcurrentHeading=77
gridSetDataAware=80
gridDataAwareFunctions=81

rem misc grid values
gridHeadingDepressedMode=1
gridHeadingNotDepressedMode=0

rem data aware functions
gridSetReadOnly$=$01$
gridDeleteRow$=$02$
gridAddRow$=$03$
gridRetrieveRow$=$04$
gridCancelUpdate$=$05$

return
```

La anterior subrutina crea una serie de variables que serán usadas por las funciones de SENDMSG() del grid, que se definen para hacer el código del programa más entendible.

Para crear la subrutina de abrir los archivos a ser usados, seleccione **New Subroutine/Function** desde el botón de Lista de **Object** y digite **Abrir Archivo,** seguidamente a las marcas de encabezado en el área de edición del Guibuilder, RESPETANDO las mayúsculas y minúsculas que se indican, digite el siguiente código:

```
abrir_archivo:
data_chan = unt
open(data_chan)"Maest_pl"

rem abre un canal alterno para realizar operaciones con el archivo
rem este canal no será atado el grid

alt_chan = unt
open(alt_chan)"Maest_pl"

Rem crea el template
datarec_desc$=""
datarec_desc$=datarec_desc$+"codigo:c(6):SHOW=1 ALIGN=0 LABEL=Código:,"
datarec_desc$=datarec_desc$+"nombre:c(30):SHOW=1 ALIGN=0 LENGTH=30 LABEL=Nombre:,"
datarec_desc$=datarec_desc$+"cedula:c(20):SHOW=1 ALIGN=0 LENGTH=20 LABEL=Cedula:,"
datarec_desc$=datarec_desc$+"direccion:c(50):SHOW=1 ALIGN=0 LENGTH=50
:LABEL=Direccion:,"
datarec_desc$=datarec_desc$+"telefono:n(10):SHOW=1 ALIGN=0 OMASK=000-00-00
:LABEL=Telefono:,"
datarec_desc$=datarec_desc$+"fecha_ingreso:n(8):SHOW=1 ALIGN=0 OMASK=00/00/0000
:LABEL=Fecha_Ingreso:,"
```

```
datarec_desc$=datarec_desc$+"tipo_planilla:c(1):SHOW=1 ALIGN=0 LENGTH=1
:LABEL=Pl:,"
datarec_desc$=datarec_desc$+"salario:n(10):SHOW=1 ALIGN=1 OMASK=#,###,##0.00
:LABEL=Salario:,"
datarec_desc$=datarec_desc$+"sexo:c(1):SHOW=1 ALIGN=0 LENGTH=1 LABEL=Sexo:"
dim datarec$:datarec_desc$
return
```

El anterior código abre al archivo con dos canales: uno para ser usado por el grid y el otro usado por el programa para la creación de nuevas llaves, también se define una descripción de template en datarec_desc$ y el template mismo datarec$: Note los atributos para cada campo, estos controlaran como se comportará y como se verán los datos en el grid

Para crear la tercer subrutina seleccione New Subroutine/Function desde el botón de Lista Object y de el siguiente nombre **Atar Grid al Canal** y OK, seguido de las marcas de encabezado en una nueva línea en el área de edición del Guibuilder digite lo siguiente:

```
Atar_grid_al_canal:

Rem obtiene el ID del grid desde el template del form
grid_id=num(fattr(datagrid_temp$,"grdmaestroGrid","ID"))

Rem envía la funcion de send message 80 (gridSetdataAware) para atar grid al canal
tf$=sendmsg(gb__sysgui,grid_id,gridSetDataAware,data_chan,datarec_desc$)
tf$=sendmsg(gb__sysgui,grid_id,gridShowCurrentHeading,gridHeadingDepressedMode,$$)

return
```

Colocando las Celdas en Modo de Edición

En el botón de Lista de Object seleccione Form 101 frmgrid
En el botón de lista de Control seleccione Push Button 201 btnEditar
En el botón de lista de Eventos seleccione Button Pushed
En el área de edición del Guibuilder digite el siguiente código que será ejecutado cada vez que el usuario presione el botón Editar

```
gosub editar_celda
```

En el Botón de lista de Control seleccione Grid 204 grdmaestroGrid
En el Botón de lista de eventos seleccione Grid Double Click
Después de las marcas de encabezado en el área de edición digite el siguiente código que será ejecutado cada vez que el usuario de doble click en una celda del grid

```
gosub editar_celda
```

Escribiendo la Subrutina de Editar celda

En el Botón de lista de Object seleccione New Subroutine/Function y digite el siguiente nombre Editar Celda y OK

Después de las marcas de encabezado en una nueva línea digite lo siguiente

```
editar_celda:
rem obtiene la fila actual
     grid_id=num(fattr(datagrid_temp$,"grdmaestroGrid","ID"))
     row$=sendmsg(gb__sysgui,grid_id,gridGetSelectedRow,0,"")
     row = dec($00$+row$)
rem obtiene la columna actual
     col$=sendmsg(gb__sysgui,grid_id,gridGetSelectedCol,0,"")
     col=dec(col$)
rem prepara la edición
     editparams_desc$="mask:c(1*=0),restore:c(1*=0),"+
:    "initstr:c(1*=0),key:u(2),col:u(2),row:u(4)"
     dim editparams$:editparams_desc$
     editparams.key = 0
     editparams.col = col
     editparams.row = row
rem bloque editando la primera columna la cual es la llave primaria
    if col=0 then
:    msg$="Usted no puede editar la llave primaria del registro.";
:    style = msgboxExclamation;
:    title$="No Edit";
:    trash=msgbox(msg$,style,title$)
:   else
:    trash$=sendmsg(gb__sysgui,grid_id,gridStartEdit,0,editparams$)
return
```

Adicionando un Nuevo Registro

En el Botón de Lista de Object seleccione Form 101 frmgrid
En el Botón de lista de Control seleccione Push Button 200 btnNuevo
En el Botón de lista de Eventos seleccione Button Pushed
En el área de edición del Guibuilder digite el siguiente código que será ejecutado cada vez que el usuario presione el botón Nuevo

```
Gosub Adicionar_registro
```

En el Botón de Lista de Object seleccione New Subroutine/Funtion y dele el nombre Adicionar registro y OK

En el área de edición de Guibuilder después de las marcas del encabezado en una nueva línea digite lo siguiente:

```
Adicionar_registro:
```

```
rem envía el mensaje de adicionar filas
      grid_id=num(fattr(datagrid_temp$,"grdmaestroGrid","ID"))
      trash$=sendmsg(gb__sysgui,grid_id,gridDataAwareFunctions,0,gridAddRow$)

rem esta bandera esta chequeada en el modo de editar del grid
rem para que el valor de la llave primaria sea puesto en el
rem programa cuando empieza a ocurrir el  evento de edición

      add_in_progress=1

return
```

Poniendo el grid en el modo de edición de eventos

En el botón de lista de Object seleccione Form 101 frmgrid
En el Botón de lista de Control seleccione Grid 204 grdmaestroGrid
En el Botón de Lista de Eventos seleccione Grid Edit Mode
En el área de edición del Guibuilder digite el siguiente código:

```
while add_in_progress
  rem muestra y edita valor en columna 0 (llave primaria)
   add_in_progress=0
   if gb__notice.col = 0 then gosub crear_nueva_llave;
:     trash$=sendmsg(gb__sysgui,gb__notice.id,gridSetEdit,0,newkey$);
:     trash$=sendmsg(gb__sysgui,gb__notice.id,gridEndEdit,0,$$);
:     desc$="mask:c(1*=0),restore:c(1*=0),initstr:c(1*=0),";
:     desc$=desc$+"key:u(2),col:u(2),row:u(4)";
:     dim editparams$:desc$;
:     editparams.key = 0;
:     editparams.col = gb__notice.col+1;
:     editparams.row = gb__notice.row;
:     trash$=sendmsg(gb__sysgui,gb__notice.id,gridStartEdit,0,editparams$)
wend
```

Creando la Subrutina de creación de nueva Llave

En el Botón de lista de Object seleccione New Subroutine/Function
De el Nombre **Crear Nueva Llave** y OK
Seguido de las marcas de Encabezado en una nueva línea digite lo siguiente:

```
Crear_nueva_llave:
rem asigna un nuevo número
      trash$=fattr(datarec$,"código")
      keylen = dec(trash$(10,2))
      keymask$ = fill(keylen,"0")
      newkey$=keyl(alt_chan,err=no_keyl)
      done=0
```

```
bump_it:
   while !(done)
      newkey$=str(num(newkey$)+1:keymask$)
      done=1
      read(alt_chan,key=newkey$,dom=got_it)
      done=0
      got_it:
   wend
return
no_keyl:
  rem esto maneja el caso de que no exista ningún registro
  newkey$=str(0:keymask$)
  goto bump_it
```

Borrando un registro

En el Botón de Lista de Object seleccione Form 101 frmgrid
En el Botón de lista de Control seleccione Push Button 202 btnBorrar
En el Botón de lista de Eventos seleccione Button Pushed
En el área de edición del GuiBuilder digite lo siguiente

```
gosub Borrar_fila_actual
```

En el Botón de lista de Object seleccione New Subroutine/Function

Y digite el nombre **Borrar Fila actual** y OK

Seguidamente a las marcas de encabezado en una nueva línea digite lo siguiente:

```
Borrar_fila_actual:
rem obtiene la columna actual
      grid_id=num(fattr(datagrid_temp$,"grdmaestroGrid","ID"))
      row$=sendmsg(gb__sysgui,grid_id,gridGetSelectedRow,0,"")
      row = dec($00$+row$)
rem crea un template temporal para tener el contenido de la fila
      dim tmp_datarec$:datarec_desc$
rem obtiene los datos de la fila actual
      tmp_datarec$=sendmsg(gb__sysgui,grid_id,gridDataAwareFunctions,row,gridRetri
      eveRow$)
rem confirma el borrado
      msg$="Esta seguro que desea Borrar el código "+tmp_datarec.código$+"?"
      style = msgboxYesNo+msgboxInfo+msgboxSecond
      title$="Confirma Exclusión"
      resp = msgbox(msg$,style,title$)
rem si la respuesta es afirmativa entonces se envía el mensaje de borrar
      if resp<>msgboxYes then
:           return
rem borra la fila
      trash$=sendmsg(gb__sysgui,grid_id,gridDataAwareFunctions,row,gridDeleteRo
:     w$)
```

```
rem desconecta el grid
      trash$=sendmsg(gb__sysgui,grid_id,gridSetDataAware,0,$$)
rem resetea el puntero del archivo
      read(data_chan,key="",err=dr_continue)
      dr_continue:
rem reconecta el grid
      trash$=sendmsg(gb__sysgui,grid_id,gridSetDataAware,data_chan,datarec_desc$)
      return
```

Terminada la digitación indicada, aún nos falta el código de eventos, pero para evitarle su digitación mejor las vamos a copiar desde otro programa ya terminado, ya que son idénticos para todo Grid. Por medio de GuiBuilder, abra el programa **c:\basis\tools\guibuild\datagrid.gbf**, busque en la caja de lista Control: ***Grid 100 grdtestGrid*** y luego busque en la caja de lista Event: los siguientes nombres ***GridEditKeyPressed*** y ***GridKeyPress***. Sin considerar los REM's del inicio, copie y pase por aparte cada block de código al programa que tenemos abierto por el otro lado, en donde el control que tenemos que seleccionar se llama ***Grid 204 grdmaestroGrid***. Cuando pase el código del evento *GridKeyPress*, debe cambiar dentro del código el nombre de estas dos etiquetas:

GOSUB delete_current_row	por → GOSUB borrar_fila_actual:
GOSUB add_record:	por → GOSUB adicionar_registro:

En el Menu Program, salve el programa y seleccione Run Program.

Manual preparado en Costa Rica por Daniel Ortíz de SPS Software <docsps@racsa.co.cr>
y Jorge Garro de UniSoft <unisoft@racsa.co.cr>

Recomendaciones para la programación con GuiBuilder

Parte de la estrategia a seguir en el uso de las **formas** o pantallas gráficas es tener cierto cuidado en la asignación de los números de identificación de los controles, jerarquizándolos y estableciendo técnicas que permitan su uso por medio del nombre que se asigna a cada control, en el momento de dibujar las pantallas con el ResBuilder.

Como primer consejo, es bueno numerar los objetos adecuadamente, porque recuerde que en el momento de correr el programa, en la entrada de datos se va saltando de un objeto a otro en el mismo orden ascendente con que los objetos están numerados. Si lo cree conveniente, puede numerarlos por ejemplo de 5 en 5 o como quiera, aunque debemos estar claros de que lo que se pretende mediante este documento, es enseñarle a trabajar en su programa con los nombres asignados a los objetos, lo cual es mucho más cómodo que usar tediosos números de identificación (ID).

Como segunda recomendación, se sugiere no programar usando el nombre largo **`GB__SYSGUI`** asignado al canal SYSGUI por el GuiBuilder. En su lugar resulta más cómodo usar un nombre más corto como **`X0`**, porque de fijo usted va a escribir menos. Si quiere seguir siendo obediente, entonces siga insertando en la sección *Initialization code* de cada programa:

```
X0=GB__SYSGUI
```

El siguiente paso, es enseñarle como utilizar en la programación los nombres asignados a las formas y a los objetos, para lo cual también puede insertar en la rutina de inicio del GuiBuilder instrucciones similares a las que siguen, para los objetos que le interese usar en el programa:

```
dim frm_screen$:fngb__template$(gb__win_id$)

Id_lboxPaises=Num(Fattr(frm_screen$,"Id_lboxPaises","ID"))
Id_btnEliminar=Num(Fattr(frm_screen$,"Id_btnEliminar","ID"))
Id_ipnNombClien=Num(Fattr(frm_screen$,"Id_inpNombClien","ID"))
Id_txtNombCia=Num(Fattr(frm_screen$,"Id_txtNombCia","ID"))
```

Para entender lo anterior, debe saber que:

- **`gb__win_id$`** contrendrá el ID de la forma a ser usada.
- **`frm_screen$`** contendrá el template de los objetos a ser usados y que
- **`Id_lboxPaises`**, **`Id_btnEliminar`** y los demás anotados, son los nombres asignados con el ResBuilder a los objetos.

Para aplicar lo anterior a un Child Window puede hacerlo así:

```
Dim frm_screen$:fngb__template$(fngb__win_id$(nfield(gb__win$,"chw_Consulta")))

Id_CodigoCliente=Num(Fattr(frm_screen$,"Id_Codigo_Cliente","ID"))
```

Lo mismo anterior pero más automático

Lo que sigue es otra forma aún mejor de obtener los nombres de todos los controles y su ID, directamente desde la Forma o Child Window que indiquemos, bajo la advertencia de que esta técnica no sirve para programas que son ejecutados por medio de la instrucción CALL, o bajo licencias RUNTIME, por hacerse uso del comando EXECUTE.

```
rem ' ------------------------------------------------------------------
rem ' Init
rem ' ------------------------------------------------------------------
      get_text=1,
:     x0=gb__sysgui

rem Carga variables con valores de la forma Cajas
   print (x0)'context'(gb__win.frm_cajas)
   dim frm_screen$:fngb__template$(fngb__win_id$((nfield(gb__win$,"frm_cajas"))))
   tpl$=fattr(frm_screen$,"")
   REPEAT
   LET A=POS($0A$=TPL$),CAMP$=FATTR(FRM_SCREEN$,TPL$(1,A-1))
   IF POS($0A$+TPL$(1,A)=$0A$+FATTR(FRM_SCREEN$,""))
:     THEN LET A$=TPL$(1,A-1)+"=NUM(FATTR(frm_screen$,"+$22$+TPL$(1,A-1)
:+$22$+","+$22$+"ID"+$22$+"))"; IF POS("LBL_"=TPL$)<>1 THEN EXECUTE A$
   LET TPL$=TPL$(A+1)
      UNTIL TPL$=""

REM Sigue abrir archivos a ser usados
///
```

Con ese block de instrucciones, de forma automática se obtienen los nombres y los valores **de todos** los ID, tal y como se nombraron con el ResBuilder. Note que se hace la exclusión de aquellos controles cuyo *name* empiece con las letras LBL_, para ignorar los objetos tipo Estatic Text, de los cuales normalmente no se requiere obtener el ID.

El ejemplo anterior aplica de manera similar para los Child Window.

Si le interesa implementarlo deberá insertar ese grupo de instrucciones en la sección *Initialization code* de cada programa, para así trabajar con el Nombre y no con el ID. Vea que todo lo anterior se hace mediante funciones que provee el GuiBuilder. Deberá recordar que es necesario repetir el block de instrucciones por cada Window o Child Window que vaya a usar el programa.

Algo sobre la función CTRL.

Sabemos que sirve para obtener la información de los controles y también posee la característica de que dentro de sus parámetros se puede mencionar un cuarto parámetro con el contexto del cual se desea extraer información, por ejemplo:

```
gb__win.frm_Pantalla
```


Manual preparado en Costa Rica por Daniel Ortíz de SPS Software <docsps@racsa.co.cr>
y Jorge Garro de UniSoft <unisoft@racsa.co.cr>

y en los demás parámetros utilizando siempre los nombres que ya sabe que provee el GuiBuilder:

```
NombCia$=CTRL(X0,Id_txtNombCia,Get_Text,gb__win.frm_Pantalla)
```

Otra forma de obtener ó almacenar información en cada control

Es utilizando funciones que también provee GuiBuilder, mediante un método muy diferente:

```
frm_Screen$=fngb__get_screen$(gb__win_id$,frm_screen$)
```

Lo anterior nos devuelve todo lo almacenado en la pantalla, es como hacer una especie de CTRL a toda la pantalla, permitiéndosenos hacer referencia a cada control por su nombre, ejemplo:

```
Print frm_Screen.Id_Documento$
Print frm_Screen.Id_Codigo$
```

Bajo este esquema también se puede actualizar o almacenar datos en cada control en la pantalla anterior, así:

```
frm_Screen.Id_Documento$="123456"
frm_Screen.Id_Codigo$="000125"
```

Hechas las modificaciones, solo queda actualizar o refrescar la información en la pantalla con la siguiente función:

```
frm_Screen$=fngb__put_screen$(gb__win_id$,frm_Screen$)
```

Lo anterior también se puede hacer de forma individual con cada campo, mediante estas otras funciones que GuiBuilder también facilita:

```
fngb__get_field$()
fngb__put_field$()
```

Para más información puede consultar el manual de ayuda en línea de Visual Pro/5.

Interacción con objetos mediante función definida por el usuario

Esta otra técnica es la del uso de una función que se DEFine en cada programa, la cual también permite utilizar los nombres de los controles en lugar de los ID o números de identificación.

Igualmente se requiere del uso de variables y funciones internas propias del GuiBuilder mencionadas anteriormente:

```
GB__WIN.NombForma
FNGB__WIN_ID$
FNGB__TEMPLATE$
```

Su DEFinición sería así:

```
DEF FNCTRL_ID(Contexto,CTRL_NAME$)
    PRINT (X0)'CONTEXT'(Contexto);
    EMPL$=FNGB__TEMPLATE$(FNGB__WIN_ID$(Contexto));
    DIM X$:TEMPL$;
    I=NUM(FATTR(X$,CTRL_NAME$,"ID"));
    RETURN I
FNEND
```

La función retorna el número de identificación de cualquier control, dándole como parámetros el Nombre de la Forma y el del Control que nos interesa. Adicionalmente, observe que mediante esta función usted se olvida de tener que estar posicionándose sobre el contexto que interesa, la función FNCTRL_ID lo hace por usted.

A continuación algunos ejemplos de cómo usarla:

- Poner un título a un 'Static Text'.
  ```
  PRINT (X0)'TITLE'(FNCTRL_ID(GB__WIN.fmDatos,"txtNombCia"),"LA EMPRESA, S.A.")
  ```

- Obtener el valor contenido en un 'Static Text'.
  ```
  NombCia$=CTRL(X0,FNCTRL_ID(GB__WIN.fmDatos,"txtNombCia"),GetText)
  ```

- Des-habilitar un botón en un Child Window.
  ```
  PRINT (X0)'DISABLE'(FNCTRL_ID(GB__WIN.fmDatos,"btnElimimar"))
  ```

- Enfocar un control InputE.
  ```
  PRINT (X0)'FOCUS'(FNCTRL_ID(GB__WIN.fmDatos,"inpNombClien"))
  ```

- Agregar un ítem a un List Box
  ```
  PRINT (X0)'LISTADD'(FNCTRL_ID(GB__WIN.fmDatos,"lboxPaises"),-1,"Costa Rica")
  ```

Note como usando los nombres de los controles, se aprecia perfectamente de cuáles controles se trata, evitándose trabajar con lo abstracto que resultan los números de identificación.

Manual preparado en Costa Rica por Daniel Ortíz de SPS Software <docsps@racsa.co.cr>
y Jorge Garro de UniSoft <unisoft@racsa.co.cr>