

# **Programación de interfases gráficas (GUI) en System RPL**

**Luis Angel Barahona Burgos**

Primera edición



# Prefacio

El presente manual es fruto del estudio de la documentación sobre el uso de las interfases gráficas en calculadoras HP y de mi propia experiencia en programación System RPL y ASSEMBLER. La programación de las GUI es muy flexible: al permitir el manejo de eventos se puede cambiar la apariencia gráfica, modificar parámetros en tiempo de ejecución, validar entradas del usuario y bloquear o modificar las opciones estándar de manejo.

La intención de este documento es facilitar la programación de las GUI y entender su modo de funcionamiento, ya que la documentación sobre el tema es limitada y se encuentran pocos ejemplos con código fuente. Además, se distribuye una serie de archivos .h con las direcciones y declaraciones `EXTERNAL` de todos los comandos y eventos expuestos en el presente manual.

Se utilizaron tres fuentes primarias de información para la elaboración de éste manual: el libro 'An Introduction to HP 48 System RPL and Assembly Language Programming' de James Donnelly, 'Programming in System RPL' de Eduardo M Kalinowski y Carsten Dominik y los archivos de encabezado incluidos en el software Debug4x. Del libro de Donnelly se obtuvo la información de parámetros de las GUI y de algunos comandos; de Kalinowski comandos adicionales no explicados por HP (Especialmente de `Choose`) y de los archivos de Debug4x los nombres de los eventos y variables locales.

Espero que la documentación sea de utilidad y que me envíen comentarios a las direcciones de correo electrónico indicadas al final.

Enero 24 de 2004

Luis Angel Barahona Burgos

# Renuncia

Éste documento tiene © de Luis Angel Barahona Burgos. Es de libre distribución, puede redistribuirlo sin limitación excepto que no se puede modificar, añadir o eliminar información sobre su contenido o sobre su autor. No se puede cobrar por su distribución excepto por el costo del soporte en el que se distribuya y los gastos de envío.

El autor entrega la documentación y demás archivos incluidos en el paquete "tal cual" y con todos los defectos. Además no existe garantía ni condición de titularidad, de posesión y disfrute plenos o de ausencia de infracción. Usted asume todo riesgo que surja de la utilización de este documento.

En ningún caso el autor será responsable por ningún daño, consecuencial, incidental, directo, indirecto especial o sancionador u otro daño cualquiera que sea que pudiere surgir del uso o imposibilidad de uso de este documento, ya sea por agravio, negligencia, responsabilidad estricta o de otra forma, aun en el caso de que se hubiera informado al autor de la posibilidad de dichos daños. Esta exclusión de daños será efectiva incluso aunque algún recurso tenga errores en su fin esencial.

Nota: Hewlett Packard, HP48G, HP49G y hp 49g+ son marcas comerciales de Hewlett Packard Company.

# Contenido

<b><u>CAPÍTULO 1.....</u></b>	<b><u>1</u></b>
<b><u>MSGBOX Y GROBS INCORPORADOS .....</u></b>	<b><u>1</u></b>
<b>1.1. DoMSGBOX.....</b>	<b>1</b>
1.1.1. EJEMPLO.....	1
<b>1.2. REFERENCIA.....</b>	<b>2</b>
<b><u>CAPÍTULO 2.....</u></b>	<b><u>3</u></b>
<b><u>BROWSER HP48 (CHOOSE) .....</u></b>	<b><u>3</u></b>
<b>2.1. MSG-HANDLER Y EVENTOS CHOOSE .....</b>	<b>3</b>
<b>2.2. ACCESO A OBJETOS DIFERENTE A LISTADOS .....</b>	<b>5</b>
<b>2.3. VARIABLES LOCALES USADAS POR CHOOSE .....</b>	<b>6</b>
<b>2.4. REFERENCIA.....</b>	<b>6</b>
<b>2.5. EJEMPLOS.....</b>	<b>9</b>
2.5.1. INDICADORES DEL SISTEMA .....	9
2.5.2. POL SALVANDO Y RESTAURANDO PANTALLA.....	10
<b><u>CAPÍTULO 3.....</u></b>	<b><u>11</u></b>
<b><u>USANDO DOINPUTFORM .....</u></b>	<b><u>11</u></b>
<b>3.1. PARÁMETROS DEL COMANDO DOINPUTFORM .....</b>	<b>11</b>
3.1.1. PARÁMETROS DE ETIQUETAS .....	11
3.1.2. PARÁMETROS DE CAMPO .....	12
3.1.3. PARÁMETROS DEL FORMULARIO.....	12
<b>3.2. MSG-HANDLER Y EVENTOS DOINPUTFORM.....</b>	<b>13</b>
<b>3.3. VARIABLES LOCALES USADAS POR DOINPUTFORM .....</b>	<b>16</b>
<b>3.4. REFERENCIA.....</b>	<b>17</b>
<b>3.5. EJEMPLOS.....</b>	<b>20</b>
3.5.1. CAMBIANDO LA APARIENCIA DEL FORMULARIO.....	20

3.5.2. FORMULARIO DE ACCESO A DATOS.....	20
--	----

**CAPÍTULO 4..... 23**

**GUÍA RÁPIDA DE USO DE DEBUG4X ..... 23**

**4.1. DESCRIPCIÓN DEL ENTORNO DE DESARROLLO DEBUG4X ..... 23**

**4.2. TIPOS DE ARCHIVO Y EL DIRECTORIO INCLUDE ..... 24**

4.2.1. EL DIRECTORIO INCLUDE ..... 24

**4.3. CREAR UN PROYECTO NUEVO ..... 25**

4.3.1. INCLUIR MÓDULOS DE CÓDIGO FUENTE ..... 26

**4.4. ESCRIBIR EL CÓDIGO FUENTE ..... 26**

4.4.1. DECLARACIONES INCLUDE ..... 27

**4.5. COMPILACIÓN Y DEPURACIÓN ..... 29**

4.5.1. PUNTOS DE INTERRUPCIÓN ..... 29

4.5.2. COMPILAR PARA HP48 Y HP49..... 29

**CAPÍTULO 5..... 31**

**CONTENIDO DEL PAQUETE PrgSysGUI ..... 31**

**5.1. HP48.H Y HP49.H..... 31**

**5.2. CARPETAS HP48 Y HP49 ..... 31**

**5.3. INSTALACIÓN DE DEFINICIONES Y DIAGRAMAS DE PILA ..... 32**

**CAPÍTULO 6..... 33**

**CONTACTÁNDOSE CON EL AUTOR..... 33**

# Capítulo 1

## MsgBox y Grobs incorporados

Las cajas de mensaje son usadas para presentar un mensaje con un dibujo opcional, esperar respuesta (Aceptar) y continuar con la ejecución del programa.

Las HP traen incorporados en ROM varios dibujos que son usados en las rutinas de interfaz gráfica y que pueden ser útiles dentro de la programación de otras aplicaciones.

### 1.1. DoMsgBox

Los parámetros requeridos por éste comando son:

<b>Nombre</b>	<b>Descripción</b>
"message"	Cadena o BINT
#maxwidth	Número máximo de caracteres por línea. Normalmente 15d
#minwidth	Número mínimo de caracteres por línea. Normalmente 10d
Grob	Gráfico para desplegar entre la caja. MINUSONE = sin gráfico
Menu	Menú

DoMsgBox siempre retorna TRUE al pulsar OK o CANCEL. Los únicos parámetros realmente personalizables son grob y Menu.

#### 1.1.1. Ejemplo

Un mensaje sencillo cambiando el menú:

```
xNAME MsgBoxSAMPLE ( -> )  
::  
  "Valores no válidos!" 15d 10d grobAlertIcon  
  { NullMenuKey NullMenuKey NullMenuKey NullMenuKey NullMenuKey  
    { "Aceptar" :: TakeOver DoMKeyOK ; } } ERBEEP DoMsgBox DROP  
;
```

No existe forma de hacer que DoMsgBox tenga salida diferente a TRUE. Lo que se puede hacer es un bucle para obligar al usuario a pulsar solo en las opciones de menú:

```
xNAME MsgBoxCicle      ( -> %1/%0)
::
  BEGIN "Salvar cambios" 15d 10d grobQueryIcon ( PTR Solo en HP39)
    { { "Si" :: TakeOver %1 TRUE 2PUTLAM ; }
      NullMenuKey NullMenuKey NullMenuKey NullMenuKey
      { "No" :: TakeOver %0 TRUE 2PUTLAM ; } }
    DoMsgBox DUPTYPEREAL? ITE TRUE DROPFALSE
  UNTIL
;
```

## 1.2. Referencia

Dirección	Nombre	Descripción
~000B1	DoMsgBox	\$message, #maxwidth, #minwidth, grob, menu → TRUE
~000B1		
~0090B1	DoMKeyOK	Salida de DoMsgBox. Retorna TRUE
^00AE002		→ TRUE
~0020B1	MsgBoxMenu	Menú por defecto con tecla OK
~0040B1		→ Menú
~850B0	grobAlertIcon	→ Grob 9x9
~0860B0		
~860B0	grobCheckKey	→ Grob 21x8 menukey
~0870B0		
~870B0	grobTitleBar	→ Grob 131x7
~0880B0		



# Capítulo 2

## Browser HP48 (Choose)

Interfaz para seleccionar uno o varios ítems en modo de ventana o pantalla completa. Es posible inicializarlo sin una lista inicial de parámetros, cambiar valores de los ítems, añadir o eliminar ítems, mostrar los ítems como dibujo, aumentar o disminuir el número de ítems desplegados, ajustar el título en ejecución, acceder a elementos diferentes a listas. Se puede tener un control absoluto sobre la visualización. `Choose` toma 5 argumentos:

<b>Nombre</b>	<b>Descripción</b>
<code>Msg-handler</code>	Rutina para el manejo de eventos
<code>TitleOb</code>	Objeto que produce la cadena para el título
<code>DecompOb</code>	Programa o BINT para convertir cada ítem en cadena. Es usado para efectuar la búsqueda por carácter.
<code>{ Choices }</code>	Listado para desplegar.
<code>#FocusPos</code>	Posición inicial para el ítem seleccionado

La salida de `Choose` es: `Ob`, `TRUE` cuando se selecciona un ítem o `FALSE` cuando se abandona la interfaz.

### 2.1. Msg-handler y eventos Choose

El manejador de eventos es un programa proporcionado por el usuario para atender los eventos de `Choose`. Al ocurrir un evento, `Choose` lo envía al manejador y si no obtiene respuesta por parte de éste, evalúa las acciones por defecto.

Cada evento es enviado al manejador como un BINT; si el manejador lo procesa debe retornar los parámetros del evento y `TRUE`, de lo contrario debe eliminar el BINT de la pila y retornar `FALSE`.

Los eventos manejados por Choose son:

<b>#msg</b>	<b>Nombre</b>	<b>Descripción</b>
57, 39	LM_SetRowCount	Número de filas mostradas por Choose → #Rows
58, 3A	LM_SetRowHeight	Alto de cada ítem en pixeles → #Height
59, 3B	LM_SetRowWidth	Ancho del ítem en pixeles (incluyendo marca CHK) → #Width
60, 3C	LM_SetViewType	FullScreen (TRUE) o modo ventana (FALSE) → TRUE/FALSE
61, 3D	LM_SetPickType	Multi-pick (TRUE) o sencillo (FALSE) → TRUE/FALSE
62, 3E	LM_SetItemCount	# total de elementos a desplegar → #ItemCount
63, 3F	LM_Set1stRowXY	Coordenadas para desplegar la ventana → #X, #Y
64, 40	LM_SetFocusPos	→ #
65, 41	LM_DispBorder	→
66, 42	LM_DispPrompt	Dibujar el título en el LCD →
67, 43	LM_SetPromptGrob	Grob del título pantalla completa / ventana → Grob w x 7
68, 44	LM_SetMaxViewPromptGrob	Grob del título pantalla completa → Grob 131 x 7
69, 45	LM_SetNormViewPromptGrob	Grob del título modo ventana → Grob w x 7
70, 46	LM_SetPromptDecomp	Cadena título → \$Prompt
74, 4A	LM_DispList	Dibujar todas las líneas del browser →
79, 4F	LM_DispItem	Dibujar el ítem especificado #Item →
80, 50	LM_SetItem	Establece el objeto del ítem #Item → Ob
81, 51	LM_SetItemGrob	Establece el grob del ítem #Item → Grob
82, 52	LM_SetItemDecomp	Establece la cadena del ítem #Item → \$Item

#msg	Nombre	Descripción
83, 53	LM_SetMenu	Definición de menu → { menu }
84, 54	LM_SaveCovWind	Salvar el LCD previo a Choose → (Salvar MenuWindow! y Window!)
85, 55	LM_PostCreate	Acción de usuario luego de creación de locales →
86, 56	LM_Pick	Acción al seleccionar un elemento #Item → #0 → (Seleccionar todos) #FFFFFF → (Borrar toda la selección)
87, 57	LM_RestCovWind	Restaurar el LCD previo a Choose → (Dibujar LCD con MenuWindow@ y Window@)
91, 5B	LM_Quit	Aceptar o impedir salida del browser con CANCEL → TRUE/FALSE
96, 60	LM_Done	Aceptar o impedir salida del browser con ENTER → TRUE/FALSE
96, 60	LM_Ok	Aceptar o impedir salida del browser con ENTER → TRUE/FALSE

Algunos eventos tienen jerarquía sobre otros; si se maneja el de más alta jerarquía el resto de eventos de ese mismo grupos es de manejo opcional. Por ejemplo, si se maneja el evento `LM_SetItemGrob`, puede no requerirse de `LM_SetItemDecomp` ya que éste es de menor jerarquía. El manejador de eventos puede generar la cadena y convertirla en grob directamente en el evento `LM_SetItemGrob`, ahorrando tiempo de ejecución.

## 2.2. Acceso a objetos diferente a listados

El comando `Choose` puede usarse para desplegar objetos diferentes a listados. Manejando los eventos `LM_SetItem`, `LM_SetItemDecomp`, `LM_SetItemGrob` y `LM_SetItemCount` es posible desplegar arreglos, texto con cambios de línea o cualquier otro objeto compuesto. El acceso a estos objetos se hace ítem por ítem, lo cual es muy benéfico en ahorro de memoria RAM cuando el objeto se encuentra en un banco de memoria y tiene un gran tamaño.

## 2.3. Variables locales usadas por Choose

El browser usa variables locales sin nombre (NULLLAM) para almacenar información de estado y parámetros:

LAM	GETLAM	PUTLAM	Descripción
1			Usada por CACHE
2	ChooseExit@	ChooseExit!	Condición de salida (TRUE=terminar)
3	DASpecFlags@	DASpecFlags!	{ } estados de pantalla antes de iniciar
4	MenuWindow@	MenuWindow!	grob 131x8 menu antes de iniciar
5	Window@	Window!	grob 131x56 LCD antes de iniciar
6	FocusPos@	FocusPos!	# Posición del ítem en el LCD
7	RowHeight@	RowHeight!	#Alto de cada ítem
8	RowWidth@	RowWidth!	#Ancho de cada ítem
9	1stRowY@	1stRowY!	Coordenada Y de la ventana
10	1stRowX@	1stRowX!	Coordenada X de la ventana
11	RowCount@	RowCount!	#Ítems desplegados en LCD
12	ItemCount@	ItemCount!	#Total de ítems
13	ChooseMenu@	ChooseMenu!	Definición de Menu
14	ViewType@	ViewType!	Choose en ventana completa?
14	IsMaxView?		
15	PickedItems@	PickedItems!	{ ítems seleccionados }
16	PickType@	PickType!	Choose multi selección?
16	IsMultiPick?		
17	ViewOnly?	ViewOnly!	Browser o solo visor?
18	LFocus@	LFocus!	#Índice
19	List@	List!	{ ítems }
20	DecompFmt@	DecompFmt!	DecompOb ::Converter
21	Prompt@	Prompt!	\$Título
22	ChooseProc@	ChooseProc!	Msg-handler

## 2.4. Referencia

Los siguientes comandos son no soportados por HP, pero son estables en la ROM 1.19-6 y 1.22. La mayoría de nombres son adoptados del libro de Eduardo Kalinowski.

Dirección	Nombre	Descripción
~0000B3	Choose	Msg-handler, TitleOb, DecompOb, {}, #Focus
~0000B3		Args Choose → Ob, TRUE / FALSE
~0050B3	ChooseMenu0	5 NullMenuKeys + OK
~0050B3		→ menú
~0060B3	ChooseMenu1	4 NullMenuKeys + CANCL + OK
~0060B3		→ menú
~0070B3	ChooseMenu2	2 NullMenu + CHK + NullMenu + CANCL+ OK
~0070B3		→ menú
~02D0B3	DoCKeyCancel	Cancela Choose (Envia LM_Quit)
^09E002		→ FALSE / Evento LM_Quit
~02B0B3	DoCKeyChAll	Chequea todos los ítems (En multiselección)
^0A0002		→
~02A0B3	DoCKeyCheck	Chequea el ítem actual (En multiselección)
^09F002		→
~02E0B3	DoCKeyOK	Evalúa la acción OK de LM_Ok
^09D002		→ Ob, TRUE / Evento LM_Ok
~02C0B3	DoCKeyUnChAll	Quita la selección de todos los ítems
^0B0002		→
~0360B3	LEDispItem	Dibuja un ítem
^0B1002		LFocus@ FocusPos@ →
~0350B3	LEDispList	Dibuja todos los ítems del LCD
^0B2002		→
~0300B3	LEDispPrompt	Dibuja el título
^0B3002		→
~0150B3	BBMoveTo	Avanza al ítem, actualiza LCD y variables
~0150B3		#ítem →
~0190B3	BBRecalOff&Disp	Recalcula el offset de la página y redibuja LCD
~0190B3		según valor de flag
		Flag →
~0220B3	BBRunEntryProc	Evalúa LM_PostCreate
~0220B3		→
~0230B3	BBReReadPageSize	Lee LM_SetRowCount y actualiza locales
~0230B3		→
~0240B3	BBReReadHeight	Lee LM_SetRowHeight y actualiza locales
~0240B3		→
~0250B3	BBReReadCoords	Lee LM_Set1StRowXY y actualiza locales
~0250B3		→
~0260B3	BBReReadWidth	Lee LM_SetRowWidth y actualiza locales
~0260B3		→
~0280B3	BBRunENTERAction	Evalúa LM_Ok
~0280B3		→ acción LM_Ok
~0290B3	BBRunCanclAction	Evalúa LM_Quit
~0290B3		→ acción LM_Quit

Dirección	Nombre	Descripción
~02F0B3	BBReDrawBackgr	Evalúa LM_DispBorder
~02F0B3		→ LM_DispBorder
~0370B3	BBGetNGrob	Evalúa LM_SetItemGrob, retorna grob
~0370B3		#ítem → grobítem
~0380B3	BBGetNStr	Evalúa LM_SetItemDecomp, retorna str
~0380B3		#ítem → \$ítem
~03B0B3	BBRereadChkEnbl	Evalúa LM_SetPickType, retorna flag
~03B0B3		→ flag_Multiselección
~03C0B3	BBRereadFullScr	Evalúa LM_SetViewType, retorna flag
~03C0B3		→ flag_PantallaCompleta
~03D0B3	BreReadMenus	Evalúa LM_SetMenu y actualiza locales
~03D0B3		→
~03E0B3	BBReReadNElems	Evalúa LM_SetItemCount y actualiza locales
~03E0B3		→
~03F0B3	BBGetN	Evalúa LM_SetItem, retorna Ob
~03F0B3		#ítem → Obítem
~04B0B3	BBIsChecked?	Retorna el estado de selección del ítem
~04B0B3		#ítem → flag
~0520B3	BBUpArrow	Grob flecha arriba
~0520B3		→ Grob 6 x 8
~0530B3	BBDownArrow	Grob flecha abajo
~0530B3		→ Grob 6 x 8
~0540B3	BBSpace	Grob espacio
~0540B3		→ Grob 6 x 8
~0590B3	BBPgDown	Adelanta página y actualiza locales sin redibujar
~0590B3		→
~05A0B3	BBPgUp	Retrocede página y actualiza locales sin redibujar
~05A0B3		→
~05B0B3	BBEmpty?	Retorna TRUE si ItemCount@ = 0
~05B0B3		→ flag
~01E0B3	DoCKeyPgUp	Retrocede página, actualiza locales y redibuja
~01E0B3		→
~01F0B3	DoCKeyPgDown	Adelanta página, actualiza locales y redibuja
~01F0B3		→
~0200B3	DoCKeyBegin	Va al inicio, actualiza locales y redibuja
~0200B3		→
~0210B3	DoCKeyEnd	Va al final, actualiza locales y redibuja
~0210B3		→
~0570B3	GetSpecFlags	Retorna un listado con los 6 indicadores del estado de pantalla antes de iniciar el browser.
~0570B3		→ { SpecFlags }
~0580B3	RestoreSpecFlags	Restaura los indicadores del estado de pantalla
~0580B3		{ SpecFlags } →

Dirección	Nombre	Descripción
~0650B3	BBStdConvert	DecompOb usado por xCHOOSE
~0610B3		ÍtemOb → \$Ítem

En el listado anterior se encuentran dos comandos útiles para la programación de interfases gráficas con `ParOuterLoop` para facilitar la restauración del entorno gráfico anterior: `GetSpecFlags` y `RestoreSpecFlags`. En la sección de ejemplos se muestra el uso de éstos comandos para restaurar pantalla de una forma limpia (sin que la ventana previa tenga que dibujar todo de nuevo).

## 2.5. Ejemplos

### 2.5.1. Indicadores del sistema

Es un ejemplo muy completo donde se usan la mayoría de eventos (`NULLNAME BrowserSysFlags`). Consiste en un browser con algunos indicadores del sistema:

```

RAD XYZ DEC R~ 'X'
Indicadores 49g+
✓ 2 Constante + Numérico
  3 Función + Simbólico
 22 Infinito = Error
✓ 35 Transferencia Binaria
✓ 36 Sobrescribir recibido
✓ 40 Mostrar Reloj
 41 Reloj 12 horas
✓ 42 Fecha dd.mm.yy +
UP DOWN ✓CHK SalirSalva

```

```

Indicadores
✓ 2 Constante + Numérico
  3 Función + Simbólico
 22 Infinito = Error
✓ 35 Transferencia Binaria
✓ 36 Sobrescribir recibido
✓ 40 Mostrar Reloj
 41 Reloj 12 horas +
UP DOWN ✓CHK SalirSalva

```

El usuario puede quitar o poner el indicador y el texto es actualizado; el evento `LM_Pick` se maneja de forma especial; al salir con `ENTER` se pregunta si se quieren guardar las modificaciones (Usando `POL_SaveChanges` del ejemplo siguiente); se utiliza ensamblado condicional para los indicadores >64 y se hace manejo de las líneas adicionales de la 49g+. El código fuente es compatible para la HP48G pero no se muestran los textos en minúscula.

## 2.5.2. POL salvando y restaurando pantalla

El siguiente ejemplo ilustra el uso de los comandos `GetSpecFlags` y `RestoreSpecFlags` en un POL con el propósito de evitar que el entorno anterior tenga que dibujar la pantalla por completo.

```

NULLNAME POL_SaveChanges ( -> FALSE / Flag, TRUE)
:: ( Uso de GetSpecFlags y RestoreSpecFlags)
  POLSaveUI ERRSET
  ::
    FALSE
    HARDBUFF ZEROZERO 131d 56d SUBGROB HARDBUFF2 TOTEMPOB
  GetSpecFlags
    {{ PSExit PWindow PSMenuWindow PSSpecFlags }}
    ' ( AppDisplay )
  ::
    DA3OK?NOTIT DispMenu
    DA1OK?NOTIT :: RECLAIMDISP ClrDA1IsStat "Guardar cambios?"
  DISPROW3 ;
    SetDA1Valid ClrDA3OK
  ;
  ' ( AppKeys )
  ::
    kpNoShift #=casedrop
    :: ( Teclas de menú, shift y ON/ENTER si están habilitadas)
    DUP#<7 casedrpfls
    kcOn ?CaseKeyDef :: TakeOver FalseTrue PSExit! ;
    kcRightShift #=casedrpfls
    DROP 'DoBadKeyT
  ;
    kpRightShift #=casedrop
    :: ( Solo habilitar la tecla de apagar y el menú)
    DUP#<7 casedrpfls
    kcRightShift #=casedrpfls
    kcOn #=casedrpfls
    DROP 'DoBadKeyT
  ;
    2DROP 'DoBadKeyT
  ;
  TrueTrue
  { { "No" :: TakeOver FalseTrue PSExit! TRUE ; }
  NullMenuKey NullMenuKey NullMenuKey NullMenuKey
  { "Si" :: TakeOver TrueTrue PSExit! TRUE ; } }
  ONEFALSE' PSExit 'ERRJMP
  POLSetUI ClrDAsOK POLKeyUI
  PWindow HARDBUFF ZEROZERO GROB!
  PSMenuWindow HARDBUFF2 ZEROZERO GROB!
  ClrDAsOK PSSpecFlags RestoreSpecFlags
  ABND
  ;
  ERRTRAP POLResUI&Err
  POLRestoreUI
  ;

```



# Capítulo 3

## Usando DoInputForm

Éste comando genera una interfaz estilo formulario de computador con controles de texto, combos y verificación. Manejando eventos es posible modificar la apariencia gráfica, desplegar gráficos en el formulario, validar valores de campos, verificar el llenado de campos y en general tener control total sobre el formulario.

Se expondrá el uso del comando `DoInputForm`, el cual está disponible en las calculadoras HP38G, HP39G, HP40G, HP48G, HP49G Y HP49G+; no confundir `DoInputForm` con `IfMain` el cual solo está disponible en la serie HP49. En una futura versión de éste manual se explicará en detalle el uso de `IfMain`.

`DoInputForm` es considerablemente más rápido en la serie HP49, por lo tanto es recomendable usar interfases sencillas en la serie HP48.

La descripción y uso de todos los eventos no está completa en la presente edición, sin embargo se listan todos los eventos, función y posible diagrama de pila.

### 3.1. Parámetros del comando DoInputForm

El comando `DoInputForm` requiere parámetros de etiquetas, de campo y de formulario:

#### 3.1.1. Parámetros de etiquetas

Cada etiqueta requiere 3 parámetros:

<b>Nombre</b>	<b>Descripción</b>
<code>Label_String</code>	Cadena o BINT con el mensaje de la etiqueta
<code>#X_Pos</code>	Coordenada X de la etiqueta
<code>#Y_Pos</code>	Coordenada Y de la etiqueta

### 3.1.2. Parámetros de campo

Cada campo necesita los siguientes 13 parámetros:

<b>Nombre</b>	<b>Descripción</b>
Fld-Handler	Manejador de eventos del campo
#X_Pos	Coordenada X del campo
#Y_Pos	Coordenada Y del campo
Fld_Width	Ancho en pixeles del campo
Fld_Height	Alto en pixeles del campo
Fld_Type	Tipo de campo: 1 Campo de texto 3 Texto en modo algebraico 12 Combo (Choose) 13 Combo-edit 32 Campo de verificación
Ob_Types	Listado especificando los tipos de objeto que acepta el campo. MINUSONE = Cualquier objeto
Decompile_Ob	Programa o BINT para convertir el campo en cadena. Al usar IFM_SetFDcomp éste parámetro no es tenido en cuenta
Fld_Help { Choices }	Cadena o BINT con el mensaje de ayuda del campo Listado para seleccionar usado en combos. Si se maneja IFM_Choose éste parámetro es irrelevante.
DecompOb	Usado solo en combos. Parámetro usado para ejecutar Choose si no se maneja IFM_Choose.
Reset_Value	Valor para los campos cuando se pulsa RESET en el menú
Init_Value	Valor inicial del campo

### 3.1.3. Parámetros del formulario

<b>Nombre</b>	<b>Descripción</b>
#LabelCount	Número de etiquetas
#FieldCount	Número de campos
Form-Handler	Manejador de eventos del formulario
Title	Cadena o BINT con el título del formulario

Todos los parámetros generan una variable local al momento de la creación del formulario; por lo tanto, es posible alterar parámetros en tiempo de ejecución como posición, tipo y en general todas las propiedades del campo y del formulario.

Los ejemplos contenidos en éste capítulo se desarrollaron con el 'Inform Box Editor' del software Debug4x, el cual provee de una interfaz gráfica para el desarrollo de formularios.

## 3.2. Msg-handler y eventos DoInputForm

El manejador de eventos es un programa proporcionado por el usuario para atender los eventos de `DoInputForm`. Al ocurrir un evento, `DoInputForm` lo envía al manejador y si no obtiene respuesta por parte de éste, evalúa las acciones por defecto.

Cada evento es enviado al manejador como un `BINT`; si el manejador de eventos lo procesa debe retornar los parámetros del evento y `TRUE`, de lo contrario debe eliminar el `BINT` de la pila y retornar `FALSE`.

`DoInputForm` cuenta con 56 eventos en total, describiéndose en los ejemplos el uso de 26 eventos que son suficientes incluso para formularios complejos. Para los restantes eventos se indica el diagrama de pila, función y nombre.

Los eventos para los cuales se desconoce el diagrama de pila o su función se listan con “???”; sin embargo, se lista el nombre del evento. Usando `Debug4x` se pueden depurar estos eventos para obtener los parámetros faltantes.

#msg	Nombre	Descripción
1, 1	IFM_DispTitle	Dibuja en pantalla el título del formulario →
2, 2	IFM_SetTitleGrob	Establece el grob del título del formualrio → Grob 131x7
3, 3	IFM_SetTitleDecomp	Establece la cadena del título del formualrio → "Titulo"
4, 4	IFM_DispClient	Dibuja en LCD todos los campos y etiquetas →
5, 5	IFM_SetClientBack	Establece el grob de etiquetas ( ClientBack) → Grob 131x39, TRUE
6, 6	IFM_DispLabel	Dibuja la etiqueta en ClientBack Grob 131x39, #Label → Grob 131x39
7, 7	IFM_DispField	Dibuja el #Fld en el LCD #Fld →
8, 8	IFM_SetIFKeyOb	Campo: ??? → ???
9, 9	IFM_SetFGrob	Establece el grob del #Fld #Fld → #Fld, grob
10, A	IFM_SetFDecomp	Establece la cadena del #Fld → \$Fld
11, B	IFM_DispHelp	Dibuja en LCD la ayuda del campo →
12, C	IFM_SetHelpGrob	Establece el grob de ayuda del campo → grob ww x 6
13, D	IFM_SetHelpDecomp	Establece la cadena de ayuda del campo → "Help"
14, E	IFM SetInitFocusFII	Establece el campo usado para foco inicial → #Fld
15, F	IFM_SetInpFormMenu	Establece el menú del formulario → {Menu}
16, 10	IFM_SetAppMenuKeys	3 teclas de la aplicación → {menu 3 teclas}, ???
17, 11	IFM_PostCreate	Acciones del usuario luego de crear el formulario →
18, 12	IFM_SetKeyOb	??? ??? → ???
19, 13	IFM_LostFocus	Acciones al perder el foco del campo ??? → ???
20, 14	IFM_GotFocus	Acciones al tomar el foco del campo MyFldID, #FocusPrev → MyFldID, #FocusPrev
21, 15	IFM_UnShowSel	Acción para des-seleccionar el campo ??? → ???
22, 16	IFM_ShowSel	Acción para seleccionar el campo ??? → ???
23, 17	IFM_EditField	Editar el campo ??? → ???

#msg	Nombre	Descripción
24, 18	IFM_Parse	Convertir cadena de edición en Ob del campo ??? → ???
24, 18	IFM_PreParse	Acción previa a la conversión del campo ??? → ???
25, 19	IFM_PostParse	Acción posterior a la conversión del campo ??? → ???
26, 1A	IFM_Choose	Generar browser y actualizar valor de campo →
27, 1B	IFM_Check	Generar evento CHK y actualizar valor del campo ??? → ???
28, 1C	IFM_Quit	Habilitar salida del formulario vía CANCL → ... Flag
29, 1D	IFM_Done	Habilitar salida del formulario vía OK → ... Flag
30, 1E	IFM_Calc	Ir a pila, traer objeto, validarlo, etc →
31, 1F	IFM_Reset	Acción para borrar valor del campo →
32, 20	IFM_SetResetChoices	Opciones REST del campo, lo llama IFM_Reset → listado reset
33, 21	IFM_ResetField	Evento de campo llamado en "Reset value" #Campo → ???
34, 22	IFM_ResetAll	Acción para borrar todos los valores del formulario llama a todos los IFM_Reset →
35, 23	IFM_DisObTypes	Evento DoInputForm ??? → ???
36, 24	IFM_SetObTypeDescs	Evento DoInputForm ??? → ???
37, 25	IFM_GetChoice	Evento DoInputForm ??? → ???
38, 26	IFM_ChooseNext	Acción al pulsar +- →
39, 27	IFM_ChooseByChr	Ejecutar búsqueda por carácter en combos Chr →
40, 28	IFM_SetChoices	Listado y foco inicial para browser en campo → {}, focus
41, 29	IFM_SetUserValue	Evento DoInputForm ??? → ???
42, 2A	IFM_SetEdLineMenu	Evento DoInputForm ??? → ???
43, 2B	IFM_SetAppEdLMenuKe	Evento DoInputForm ??? → ???
44, 2C	IFM_EndEdit&STO	Evento DoInputForm ??? → ???

#msg	Nombre	Descripción
45, 2D	IFM_ChkObType	Evento DolInputForm ??? → ???
46, 2E	IFM_ChkObValue	Verificar el valor del campo FldVal → Flag
47, 2F	IFM_PostChange	Acción luego de modificar valor de campo →
48, 30	IFM_SetDownFID	#Foco siguiente al campo actual → #FldFocusDown
49, 31	IFM_SetUpFID	#Foco anterior al campo actual → #FldFocusUP
50, 32	IFM_SetFARDownFID	Evento DolInputForm ??? → ???
51, 33	IFM_SetFARUpFID	Evento DolInputForm ??? → ???
52, 34	IFM_SetNextFID	#Foco siguiente al campo actual → #FldFocusNext
53, 35	IFM_SetPrevFID	#Foco anterior al campo actual → #FldFocusPrev
54, 36	IFM_SetNextFocusFII	Evento DolInputForm ??? → ???
55, 37	IFM_SetFocusOk?	Especifica si el campo debe tener foco → Flag
56, 38	IFM_PostDone	Evento DolInputForm ??? → ???

### 3.3. Variables locales usadas por DolInputForm

LAM	GETLAM	PUTLAM	Descripción
1			Usada por CACHE
2	GetFieldOffset	SetFieldOffset	Típicamente 55d
3	GetLabelOffset	SetLabelOffset	Típicamente 64d
4	GetMyFieldId	SetMyFieldId	#Actual de campo
5	GetFocus	SetFocus	#Campo con el foco
6	GetIFKeyOb	SetIFKeyOb	
7	GetIFMenuRow	SetIFMenuRow	# primer elemento del menu
8	GetIFMenu	SetIFMenu	Menú del formulario
9	GetFAreaStates	SetFAreaStates	
10	GetClientBack	SetClientBack	Grob 131 x 39
11	GetIFExit	SetIFExit	Condición de salida del formulario
12	GetTitle	SetTitle	Parámetro Title del formulario
13	GetFormProc	SetFormProc	Form-handler
14	GetFieldCount	SetFieldCount	#de campos
15	GetLabelCount	SetLabelCount	#de etiquetas

## 3.4. Referencia

### Comandos generales

Dirección	Nombre	Descripción
~0F0B0	IFEDispTitle	Dibuja el título en la pantalla
~0F0B0		→
~100B0	IFEGetTitleGrob	Retorna el grob del título del formulario
~100B0		→ grob 131 x 7
~110B0	IFEGetTitleDecomp	Retorna cadena del título del formulario
~110B0		→ \$Title
~120B0	IFEGetGrobClientLb	Dibuja #Label en ClientBack
~120B0		Grob 131x39, #Label → Grob 131x39
~120B0	IFEDrawLabel	Dibuja #Label en ClientBack
~120B0		Grob 131x39, #Label → Grob 131x39
~0E0B0	IFEDrawClientBack	Dibuja en ClientBack todas las etiquetas
~0E0B0		→ Grob 131x39, True
~0C0B0	IFEDispClientFull	Dibuja etiquetas y campos en LCD
~0C0B0		??? → ???
~130B0	IFEDispField	Dibuja el #Fld en LCD
^00BC002		#Fld →
~050B0	IFMenuRow1	EDIT + CHOOSE + CHK
~050B0		→ {Menú}
~060B0	IFMenuRow2	RESET + CALC + TYPES + NULL + CANCL + OK
~060B0		→ {Menú}
~560B0	DoKeyEdit	Ejecuta IFM_EditField
^00B5002		→
~570B0	DoKeyChoose	Ejecuta IFM_Chose
^00B6002		→
~580B0	DoKeyCheck	Ejecuta IFM_Check
^00B7002		→
~590B0	DoKeyCancel	Ejecuta IFM_Quit
^00AF002		→ FALSE
~5A0B0	DoKeyOK	Ejecuta IFM_Done
^00B4002		→ Ob1 ... Obn TRUE
~5B0B0	DoKeyReset	Ejecuta IFM_Reset
^00B8002		→
~5C0B0	DoKeyCalc	Ejecuta IFM_Calc
^00B9002		→
~5D0B0	DoKeyTypes	Ejecuta IFM_DisObTypes
^00BA002		?? → ??
~450B0	IFEGetNStr	Campo como cadena via IFM_SetFDecomp
~450B0		#Fld → \$Fld

## Comandos para leer o establecer propiedades de etiqueta

Dirección	Nombre	Descripción
~B00B0	gLblDecomp	Retorna la cadena de la etiqueta
~B30B0		#Lbl → \$Lbl
~B10B0	gLblXpos	Coordenada X de la etiqueta
~B40B0		#Lbl → #Xpos
~B20B0	gLblYpos	Coordenada Y de la etiqueta
~B50B0		#Lbl → #Ypos
~B30B0	gLblXYpos	Coordenadas X Y de la etiqueta
~B60B0		#Lbl → #Xpos, #Ypos
~B40B0	sLblDecomp	Sobreescribe la etiqueta
~B70B0		\$Lbl, #Lbl →
~B50B0	sLblXpos	Sobreescribe coord. X de la etiqueta
~B80B0		#Xpos, #Lbl →
~B60B0	sLblYpos	Sobreescribe coord. Y de la etiqueta
~B90B0		#Ypos, #Lbl →
~B70B0	sLblXYpos	Sobreescribe coord. X Y de la etiqueta
~BA0B0		#Xpos, #Ypos, #Lbl →

## Acceso a propiedades de campo:

Dirección	Nombre	Descripción
~B90B0	gFldProc	Fld-handler del campo
~BC0B0		#Fld → Fld-handler
~BA0B0	gFldXPos	Coordenada X del campo
~BD0B0		#Fld → #X_Pos
~BB0B0	gFldYPos	Coordenada Y del campo
~BE0B0		#Fld → #Y_Pos
~BC0B0	gFldWidth	Ancho en pixeles de campo
~BF0B0		#Fld → #Fld_Width
~BD0B0	gFldHeight	Alto en pixeles de campo
~C00B0		#Fld → #Fld_Height
~BE0B0	gFldType	BINT especificando el tipo de campo
~C10B0		#Fld → #Fld_Type
~BF0B0	gFldObTypes	Tipos de objetos adminitos por el campo
~C20B0		#Fld → Ob_Types
~C00B0	gFldDecompOb	Conversor Valor - cadena del campo
~C30B0		#Fld → Decompile_Ob



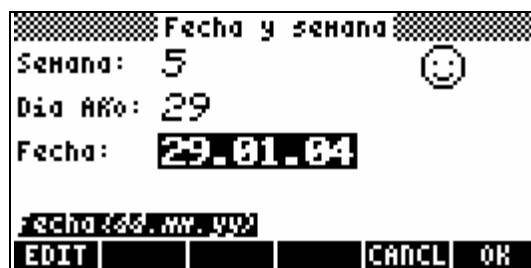
Dirección	Nombre	Descripción
~C10B0	gFldHelp	Cadena o BINT de ayuda del campo
~C40B0		#Fld → #Help/\$Help
~C20B0	gFldChooseData	Listado de selección
~C50B0		#Fld → {Choices}
~C30B0	gFldChDcmpFmt	Convertor Valor - cadena para selección
~C60B0		#Fld → Decomp_Ob
~C40B0	gResetValue	Valor por defecto del campo
~C70B0		#Fld → Reset_Value
~C50B0	gFldVal	Valor actual del campo
~C80B0		#Fld → Value
~C90B0	sFldProc	Fld-handler del campo
~CC0B0		Fld-Handler, #Fld →
~CA0B0	sFldXPos	Coordenada X del campo
~CD0B0		#X_Pos, #Fld →
~CB0B0	sFldYPos	Coordenada Y del campo
~CE0B0		#Y_Pos, #Fld →
~CC0B0	sFldWidth	Ancho en pixeles de campo
~CF0B0		#Fld_Width, #Fld →
~CD0B0	sFldHeight	Alto en pixeles de campo
~D00B0		#Fld_Height, #Fld →
~CE0B0	sFldType	BINT especificando el tipo de campo
~D10B0		#Fld_Type, #Fld →
~CF0B0	sFldObTypes	Tipos de objetos adminitos por el campo
~D20B0		Ob_Types, #Fld →
~D00B0	sFldDecompOb	Convertor Valor - cadena del campo
~D30B0		Decompile_Ob, #Fld →
~D10B0	sFldHelp	Cadena o BINT de ayuda del campo
~D40B0		\$Help, #Fld →
~D20B0	sFldChooseData	Listado de selección
~D50B0		{Chices}, #Fld →
~D30B0	sFldChDcmpFmt	Convertor Valor - cadena para selección
~D60B0		Decomp_Ob, #Fld →
~D40B0	sResetValue	Valor por defecto del campo
~D70B0		Reset_Value, #Fld →
~D50B0	sFldVal	Valor actual del campo
~D80B0		Value, #Fld →
~C80B0	GetFieldVals	Retorna valor de todos los campos
^00BB002		→ FldVal1 ... FldValn
~D80B0	PutFieldsVals	Establece valores para todos los campos
~DB0B0		FldVal1 ... FldValn →

## 3.5. Ejemplos

A continuación se describirán ejemplos del manejo de eventos en DoInputForm. El código fuente puede ser examinado en los archivos .s o haciendo doble clic en el módulo correspondiente en la ventana de proyecto de Debug4x.

### 3.5.1. Cambiando la apariencia del formulario

En este ejemplo se manejan eventos para cambiar la apariencia gráfica del formulario: insertar un grob sobre el formulario, mostrar el texto de los campos con varios formatos, cadena de ayuda en grob inverso y quitar el foco a los campos semana y día del año.



```

+-----+ Fecha y semana +-----+
Semana: 5
Dia Año: 29
Fecha: 29.01.04
Fecha (dd.mm.yy)
EDIT CANCL OK

```

También se hace validación de rango en el campo fecha para asegurar que el usuario introduce fechas válidas; la fecha es introducida como un número según la configuración del sistema (dd.mmyyyy ó mm.dyyyyy), el formulario valida el valor del número introducido, calcula la semana y día del año actualizando los valores de los otros campos. En el campo fecha, la opción de menú Reset es inhabilitada para evitar cargar valores por defecto.

### 3.5.2. Formulario de acceso a datos

En éste ejemplo se usa un formulario para acceder a datos de una estructura de base de datos (formada por listados), con la posibilidad de hacer adición, edición y borrado de registros.

Adicional a la programación del formulario, se desarrollaron una serie de rutinas preliminares para el acceso a datos, las cuales permiten el cargue del formulario, guardado de registros, navegación, creación y edición.



La aplicación accede a una base de datos de tuberías en PVC. Las tuberías se clasifican según el RDE (Relación diámetro espesor), el cual especifica la presión máxima de trabajo. Cada RDE tiene asociados una serie de diámetros disponibles comercialmente y según el RDE, un único diámetro interno que es usado con fines de diseño hidráulico.

## Modelo de base de datos

La aplicación requiere una carpeta en HOME llamada PVC.DA y una variable en este directorio con el nombre RDETable conteniendo una lista vacía. RDETable es un listado compuesto por dos listas: la primer lista contiene los números de RDE y la segunda la presión de trabajo de cada RDE. Por ejemplo: { { 21. 26. } { 200. 160. } } especifica que existen los RDE 21 y 26 con presiones de trabajo de 200 y 160 psi respectivamente.

Al crear un nuevo RDE, la aplicación crea una variable llamada RDExxTable (xx indica el número del RDE correspondiente), la cual es un listado de los diámetros asociados al RDE. Cada registro de esta variable es una lista con 4 valores: diámetro comercial en pulgadas, diámetro comercial en milímetros, diámetro externo y diámetro interno.

## Descripción del formulario

El campo RDE actualiza el campo Presión usando RDETable e indica la tabla RDExxTable para actualizar los cuatro campos restantes. En el título del formulario se muestra el número de registro actual y el total de registros del RDE.

En el menú se encuentran las funciones de navegación: Adicionar, borrar, siguiente y anterior. Al salir del formulario, avanzar o retroceder, los datos son validados y guardados en la tabla correspondiente.

## Programación del formulario

A groso modo, se debe tener en cuenta:

- Crear el entorno de variables locales de acceso a datos.
- En el momento de cargue del formulario buscar valores almacenados en PVC.DA, establecer el valor de los campos y actualizar el origen de datos.
- Administrar el evento PostChange del campo RDE para actualizar el campo Presión y establecer si es necesario, el nuevo origen de datos para los cuatro campos restantes.
- Manejar el evento Choose del campo RDE con la finalidad de actualizar el campo presión y el origen de datos.
- Crear una rutina para validar que el total de los campos del formulario tengan datos y de ésta forma, habilitar el guardado de la información.
- Establecer el menú del formulario con los botones de acceso a datos: adicionar, borrar, siguiente y anterior.
- Validar el rango de valores en general para todos los campos.

En el código fuente (Form\_Data.s) se explica con más detalle todos los aspectos de la programación del formulario y en DA\_Functions.S las funciones y variables locales usadas para el acceso a datos.

Las funciones para el acceso a datos pueden ser mejoradas o adaptadas para manejar otros objetos como arreglos, strings, etc... La intención de éste ejemplo es mostrar la funcionalidad de los formularios para ser enlazados directamente a estructuras de base de datos; no se pretende que los comandos sean un estándar para el manejo de datos, simplemente es un ejemplo de la potencialidad de los formularios.

# Capítulo 4

## Guía rápida de uso de Debug4x

### 4.1. Descripción del entorno de desarrollo Debug4x

Debug4x es un kit para el desarrollo de aplicaciones para calculadoras HP. La versión actual es la 2.1 y fue desarrollada por William G. Graves.

El presente manual no pretende ser una guía completa de uso de Debug4x (para eso está la documentación correspondiente), sin embargo, se explicarán algunos aspectos no documentados para poder trabajar adecuadamente con las definiciones y ejemplos incluidos.

Las principales características de éste software son:

- Completo soporte para las calculadoras HP48 y HP49 (También se puede usar para desarrollo de aplicaciones para la HP38, HP39 y HP40 ajustando el objeto binario de salida)
- Automatiza la generación de directorios y librerías.
- Editor de formularios basados en `IfMain` y `DoInputForm`.
- Integración con EMU48 para depuración y pruebas de las aplicaciones generadas. Los archivos son automáticamente cargados e instalados en el emulador para los procesos de depuración.
- Depuración gráfica con puntos de interrupción, visualización de pila, variables locales, gráficos y otros parámetros de la calculadora.

- Editor con sintaxis en colores, autocompletación (ctrl.+SPC), diagrama de pila de comandos (ctrl.+J), identificación de rutinas (ctrl.+Shift+B) de código RPL y listados y otras características adicionales.
- Soporte completo para depuración en RPL y ensamblador SATURN.

## 4.2. Tipos de archivo y el directorio Include

Debug4x reconoce los siguientes tipos de archivo:

- Proyectos Debug4x (\*.hpp) Los archivos de éste tipo contienen información sobre el tipo de proyecto (objeto, directorio o librería), emulador usado, archivos de código fuente e información sobre los archivos en edición.
- Código fuente (\*.s) Contienen el código fuente en ASSEMBLER o RPL. Pueden ser módulos (Archivo con varias declaraciones NULLNAME o xNAME) o fuente incluido (sin estas declaraciones).
- Definiciones (\*.h) Contienen definiciones usadas por los módulos del proyecto (llamadas de funciones entre módulos) y definiciones propias para el proyecto en particular.

Los archivos \*.E48 y \*.E49 son usados para la depuración del código. Cada proyecto puede tener su propia sesión de emulador con contenidos RAM diferentes.

### 4.2.1. El directorio Include

Es una carpeta ubicada dentro de la carpeta de Debug4x. Contiene los siguientes archivos:

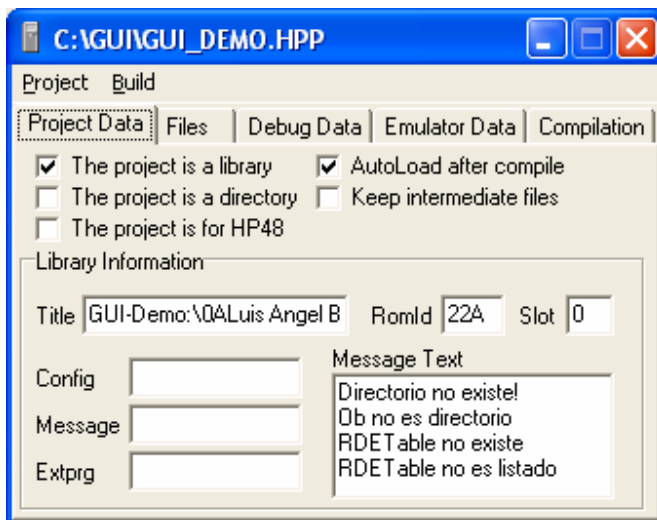
- Suprom48.a y Suprom49.a: Contiene los nombres y direcciones de los comandos usados por la HP48 y HP49 respectivamente.
- Suprom48.stk y Suprom49.stk: Nombres de comandos con su respectivo diagrama de pila. Estos archivos se pueden modificar para añadir nuevos diagramas de pila.

- `InformBox.h` e `InformBox48.h`: Definiciones usadas por el editor de formularios. Son incluidos automáticamente al crear un formulario.
- `Header.h`: Definiciones estándar. Este archivo es incluido automáticamente al crear un nuevo formulario, pero entra en conflicto con las definiciones dadas en este manual. Cada proyecto puede tener su propio `Header.h` evitando así usar el archivo del directorio `Include`; típicamente el contenido de este archivo será `INCLUDE MyProject_Header.h`. Más adelante se detallará el uso de los archivos `*.h`.

Los archivos `*.h` que sean copiados en ésta carpeta pueden ser usados en cualquier proyecto y entran a ser parte de los comandos habituales de programación.

### 4.3. Crear un proyecto nuevo

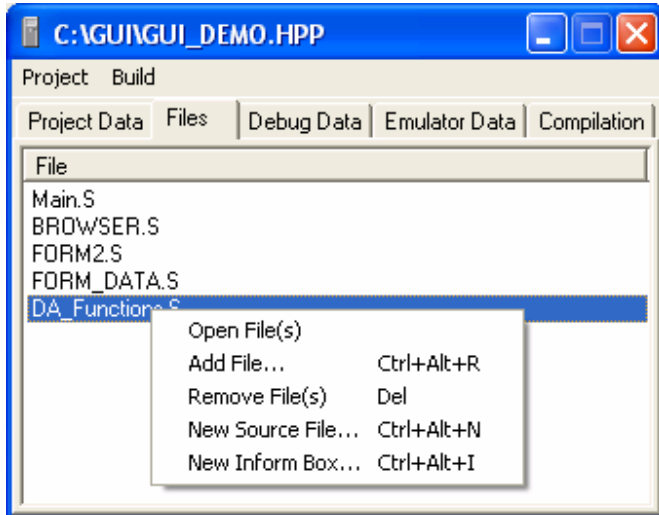
Ejecute `Debug4x.exe` y seleccione "New Project..." en el menú "Project"; establezca el nombre de archivo y guarde.



Habilitando la opción `AutoLoad` after compile la librería es automáticamente cargada e instalada en el emulador, habilitando el uso de puntos de interrupción.

### 4.3.1. Incluir módulos de código fuente

En la ventana del proyecto seleccione la ficha Files y usando el menú contextual seleccione `New Source File...` (Módulo de código) o `New Inform Box...` (Módulo de formulario).



Los archivos mostrados en esta lista son módulos de código fuente. Cada módulo debe tener los archivos \*.h necesarios para comunicarse entre ellos.

Los formularios son tratados como módulos.

## 4.4. Escribir el código fuente

Debug4x proporciona un completo editor de código fuente. Pueden editarse módulos, archivos incluidos y en general, cualquier tipo de archivo.

La siguiente figura muestra el aspecto de la ventana de edición con parte del código del módulo `Main.s`. El contenido de este módulo se utilizará para explicar algunos detalles de los procesos de programación, depuración y compilación.



```

1 RPL
2 ( C:\HP48\LIBRE\GUI\Main.S, part of the GUI_DEMO.HPP project, c
3
4 INCLUDE GUI_DEMO.h      ( Definiciones NULLNAME y xNAME)
5 INCLUDE GUI_Header.h   ( Definiciones del proyecto)
6
7 xNAME PruebasGUI
8 ::
9 grobAlertIcon
10 grobCheckKey
11 grobTitleBar
12 ;
13 xNAME FormDateTime      FormDateTime ( Comando de otro modulo)
14 xNAME MsgBoxSAMPLE     ( -> )
15 ::
16 "Valores no válidos!" 15d 10d grobAlertIcon
17 ( NullMenuKey NullMenuKey NullMenuKey
18   NullMenuKey NullMenuKey
19   ( "Aceptar" :: TakeOver DomKeyOK ; ) )
20 ERRBEEP DoMsgBox DROP
21 ;
22 xNAME MsgBoxCicle      ( -> %1/%0) :: ;
23 INCLUDE MsgBoxCicle.s  ( Fuente incluido)
24 *****
25 NULLNAME InfoGUI
26 ::
27 "Ejemplo de GUI V 1.0 Luis A. Barahona. 2004" 15d 10d
28 MINUSONE MsgBoxMenu DoMsgBox DROP
29 ;

```

Line 23    Start 0EE0B6    Stop 0EE772    adr from line 0EE1D2 ASM    CR+LF (Windows)

#### 4.4.1. Declaraciones INCLUDE

La sentencia `INCLUDE` incluye el contenido del archivo como parte del módulo o fuente donde se haga el llamado. Generalmente se usa `INCLUDE` con archivos `*.h` y `*.s`

La línea `INCLUDE GUI_DEMO.h` proporciona las definiciones de los comandos usados en todos los módulos y sus declaraciones `EXTERNAL` (Compilar como `ROMPTR`). El archivo `GUI_DEMO.h` es actualizado automáticamente por Debug4x al añadir o quitar comandos de cualquier módulo.

La línea `INCLUDE GUI_Header.h` proporciona las definiciones y direcciones de comandos adicionales usados solo en el proyecto actual. Por ejemplo, para evitar usar `LAM Ind` se puede hacer la declaración:

```
DEFINE DA_Index      LAM Ind
```

En los ejemplos proporcionados todos los módulos deben incluir los archivos `GUI_DEMO.h` y `GUI_Header.h` para que al compilar sean reconocidos los comandos de los módulos y las nuevas entradas.

Usar `DEFINE` es útil para facilitar el entendimiento del código, pudiendo usar nombres largos en el fuente y compilarlos en un menor tamaño. La declaración `DEFINE` no genera automáticamente entrada para autocompletación, si se quiere habilitar ésta característica se debe hacer lo siguiente:

```
ASSEMBLE
*Generar entrada para autocompletación con dirección 0:
=DA_Index          EQU 0      *Compila como PTR 0
RPL
*Re-definir la entrada para que no compile como PTR 0:
DEFINE DA_Index          LAM Ind
DEFINE DA_Index!        ' LAM Ind STOLAM
```

Usando éste método se consigue lo siguiente: `DA_Index` compila como `LAM Ind` pudiendo usar autocompletación y `DA_Index!` Compila como `' LAM Ind STOLAM` pero sin autocompletación (no es problema, solo se añade `!` con la primer entrada).

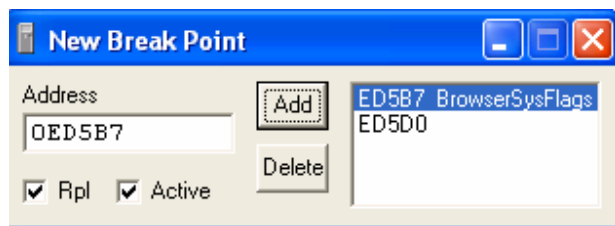
La sentencia `INCLUDE` también es usada para incluir código fuente dentro del módulo. La línea `INCLUDE MsgBoxCicle.s` incluye el contenido de `MsgBoxCicle.s` dentro del archivo actual. Esta estrategia se usa para hacer más claro el código o para incluir otros objetos (Arreglos, grobs, etc) de los cuales no necesitemos ver siempre el código fuente.

## 4.5. Compilación y depuración

En la ficha 'Emulator Data' de la ventana de proyecto seleccione los archivos del emulador a utilizar, habilite la opción 'Autoload after compile' en la ficha 'Project Data', abra el emulador (Ctrl+Alt+E) y a continuación compile el proyecto (F9); el resultado será cargado e instalado en el emulador.

### 4.5.1. Puntos de interrupción

Permiten detener la ejecución del programa en un punto específico para iniciar la depuración paso a paso. Los puntos de interrupción (Break Point) se establecen en el editor haciendo clic en la parte izquierda de la línea deseada pero solo en los módulos.



Al hacer doble clic sobre la dirección, se muestra el punto en el editor. Los puntos de interrupción solo están disponibles cuando el emulador está en ejecución. Si el emulador o el proyecto son cerrados, los puntos de interrupción desaparecerán.

### 4.5.2. Compilar para HP48 y HP49

En general, las calculadoras gráficas HP tienen bastante compatibilidad al nivel de código fuente. Debug4x permite cambiar fácilmente el objeto compilado para éstos dos modelos de calculadora.

Al habilitar la opción 'The Project is for HP48' en la ficha 'Project Data' de la ventana de proyecto, se establece que la salida se debe compilar para una HP48. Sin embargo, puede que esto no sea suficiente para compilar correctamente, ya que el mismo comando puede ser ROMPTR en la HP48 y FLASHPTR en la HP49.

En el proyecto ejemplo todos los módulos tienen incluido el archivo `GUI_DEMO.h` en el cual se hacen definiciones adicionales. Dentro de dicho archivo también se incluye `ADDRPL\HP49.H` cambiando este archivo por `HP48.h`, el proyecto compilará sin errores para la HP48.

Además, se puede hacer uso del ensamblado condicional para generar código para un modelo determinado de calculadora. Ésta técnica se usa dentro del módulo `Main.s` para convertir cadenas de caracteres con formato en objetos gráficos.

La HP49 soporta directamente la conversión con el comando `$>grob` pero la HP48 no posee ésta característica. Si en el código queremos un comando que nos convierta una cadena de caracteres en un grob con formato en cursiva, se puede hacer lo siguiente:

```

NULLNAME $>grobCur ( BINT/str -> grob)
:: ( convierte BINT o Str en grob con formato cursivo:)
DUPTYPECSTR? NOT_IT JstGETTHEMSG
* Condición de ensamblado:
ASSEMBLE
    IF (HP49) ! (HPARM)      * HP49 OR ARM
RPL
*Fuente para la HP49 o HP48gII:
    $ "\13\02\13" DUPROT &$ SWAP&$ $>grob
ASSEMBLE
    ELSE      . *La HP48G no maneja estilos en $>grob
RPL
* En la HP48G se puede hacer programa usando FNT1 y SGROB
*de Jack Levy
    $>grob
ASSEMBLE
    ENDIF
RPL
;
```

El código anterior da formato cursivo a la cadena de entrada y usa `$>grob` para la HP49 y HP48gII; para la HP48G simplemente no genera formato.

En los archivos `HP48.h` y `HP49.h` se establecen las banderas necesarias para hacer el ensamblado condicional (en éste caso HP49 y HPARM). El ensamblado condicional también se puede usar para crear varias versiones del programa, por ejemplo: demo, estándar y profesional sin tener que hacer un proyecto por cada versión.

# Capítulo 5

## Contenido del paquete PrgSysGUI

Este manual se distribuye junto a una serie de archivos en una carpeta llamada ADDRPL. Se encuentra lo siguiente:

### 5.1. HP48.h y HP49.h

Contiene definiciones de banderas para ensamblado condicional y declaraciones INCLUDE de archivos específicos para cada modelo de calculadora: se incluyen entradas no soportadas (UEP), definiciones de teclado, números en base decimal, punteros RAM y comandos y eventos para uso de formularios y browser.

Estos archivos se incluyen típicamente en el archivo de definiciones del proyecto (Para el proyecto ejemplo GUI\_Header.h).

### 5.2. Carpetas HP48 y HP49

Contienen varios archivos con definiciones para la HP48 y la HP49. Al incluir el archivo HP48.h o HP49.h dentro de los módulos del proyecto se tiene acceso a nuevos comandos, todo con la característica de autocompletación.

En el caso de la HP48 tenemos:

- BINTS48.h Entradas para los números en formato decimal. Por ejemplo, la entrada LISTCMP puede ser reemplazada por la entrada 82d, lo cual puede hacer más claro el código la mayoría de las aplicaciones.
- Choose48.h Comandos y eventos para manejo del comando Choose.
- Entradas48.H Entradas varias no soportadas y probablemente inestables con cambios de ROM. Se debe probar antes de usarlas definitivamente.

- IForm48.h Comandos y eventos para manejo del comando DoInputForm.
- KeyDefs48.h Definiciones de teclas en formato kc+KeyName. Son muy útiles para diseñar interfaces gráficas compatibles para varias calculadoras, incluso con cambios en la distribución y numeración del teclado.
- MsgBoxAndGrob48.h Comandos para manejo de DoMsgBox y objetos gráficos de la HP.
- RAM48.h Punteros RAM (Versión muy preliminar e incompleta).

Todos estos archivos son modificables. Para añadir más entradas hay que tener en cuenta que se debe trabajar directamente con la dirección del comando y no con un mnemónico; además, las entradas ROMPTR y FLASHPTR deben acompañarse de una declaración EXTERNAL o FEXTERNAL respectivamente.

### 5.3. Instalación de definiciones y diagramas de pila

Para disponer de todos los comandos y eventos mencionados es este manual, siga los siguientes pasos:

1. Pegue la carpeta ADDRPL en la carpeta Include de Debug4x. Al hacer esto junto a la declaración INCLUDE ADDRPL\HP48.h, podrá usar todos los comandos explicados (Previamente necesitará compilar el proyecto para generar la autocompletación).
2. Pegue los archivos Suprom48.stk, Suprom49.stk en la carpeta Include de Debug4x. También puede abrir los archivos Suprom48.stk, Suprom49.stk y GUI.stk con el editor de Debug4x y pegar el contenido de GUI.stk en los dos primeros archivos. Esto habilitará los diagramas de pila para los comandos descritos en este manual.

**Importante:** Debug4x tiene un pequeño BUG con la autocompletación: Si el comando está seguido por un tabulador, al ejecutar Ctrl+J no muestra el diagrama; una solución inicial es escribir el comando, luego espacio y la tabulación.

Ya que la carpeta ADDRPL queda dentro la carpeta Include de Debug4x, puede usar las nuevas definiciones en cualquier proyecto.

## Capítulo 6

# Contactándose con el autor

Para sugerencias, dudas y comentarios acerca de este manual, puede contactarse vía E-Mail con Luis Ángel Barahona Burgos a las siguientes direcciones:

- [LBBurgos2@Yahoo.com](mailto:LBBurgos2@Yahoo.com)
- [LBBurgos@Eudoramail.com](mailto:LBBurgos@Eudoramail.com)

O mediante el foro de adictosHP (<http://www.adictoshp.org>) bajo el nombre LBurgos.