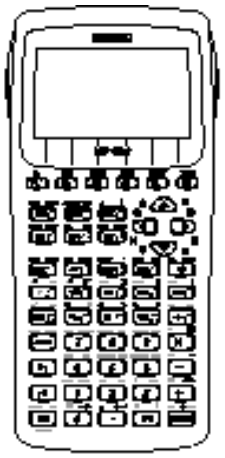
HP 49G

MANUAL DE PROGRAMACION EN SYSTEM RPL



HEWLETT PACKARD

INDICE

1 - INTRODUCCIÓN 2
2 - MI PRIMER PROGRAMA EN SYSTEM 3
3 - ALGUNAS IGUALDADES DE LOS COMANDOS DE USER 4
4 - TIPOS DE OBJETOS 5
5 - CONVERSIÓN DE OBJETOS 8
6 - ALGUNOS TIPOS DE OBJETOS 10
7 - FUNCIONES DE COMPARACIÓN O TEST 11
8 - FUNCIONES LÓGICAS 12
8.1 - OTROS COMANDOS DE TESTS 13
9 -VERIFICAR EL TIPO DE ARGUMENTO 15
9.1 - NUMERO DE ARGUMENTOS 15
10 - INDICADORES DE SISTEMA 17
11 - OPERACIONES DE LA PILA 18
12 - UTILIDADES ADICIONALES DE MEMORIA 23
13 - COMANDOS DEL SISTEMA 24
13 - FUNCIONES DE NUMEROS COMPLEJOS 29
14 - NUMEROS REALES 31
14.1 - FUNCIONES DE NUMEROS REALES 31
15 - MANIPULACION DE ENTEROS BINARIOS 37
16 - ENTRADA DE DATOS 38
17 - InputLine 40
18 - COMANDOS DE PROGRAMACION 43
18.1 - PALABRAS PROPORCIONADAS AL BUCLE DO 44
18.2 - BUCLES INDEFINIDOS 48
19 - VARIABLES TEMPORALES 50
20 - VARIABLES Y DIRECTORIOS 54
20.1 - OTRAS PALABRAS RELACIONADAS 56
20.2 - PALABRAS PARA LA MANIPULACIÓN DE DIRECTORIOS 57
20.3 - EL DIRECTORIO OCULTO 59
21- OBJETOS DE UNIDADES 60
22 - OBJETOS ETIQUETADOS 64
22 - HERRAMIENTAS DE ERROR 65
23 - LA PANTALLA 67
23.1 - GRAFICOS Y PANTALLA 68
23.2 - COMO DESPLAZAR UN GROB 76
24 - MENUS 77
25 - BACKUPS 80
26 - DoImputForm 81
27 - COMANDOS QUE OPERAN EN UN PROGRAMA 86
28 - MENSAJES TEMPORALES 89
29 - UBICACION DE LAS TECLAS 90
29.1 - ESPERAR UNA TECLA 91
30 - EL BUCLE EXTERNO PARAMETRIZADO.. 92
30.1 UTILIDADES DEL BUCLE EXTERNO PARAMETRIZADO 93
30.2 EXAMEN DEL BUCLE EXTERNO PARAMETRIZADO 94
30.3 CONTROLAR ERRORES CON LAS UTILIDADES 95
30.4 ASIGNACIONES DE TECLAS FISICAS 96
30.5 ASIGNACIONES DE TECLAS DE MENÚS 97
30.6 EVITAR ENTORNOS SUSPENDIDOS 98
31 - FILER49 101
32 - EDITOR DE LA HP49G 105
33 - HP49 BROWSER 115
34 - HP48 BROWSER 121
35 - DEFINE INCLOB INCLUDE 131
EJEMPLOS SYSTEM RPL 133

SYSTEM RPL

PARA LA HP49G

1- INTRODUCCION

El programar en System resulta mas peligroso pero a la ves mas rápido, mas peligroso por que si se introduce un dato incorrecto puede ocasionar la perdida de la memoria ya que este lenguaje no verifica los argumentos.

Por ejemplo si en UserRPL nosotros ejecutamos **«+»** y si no hay ningún argumento en la pila este mandara un mensaje de error, pero en SysRPL si ejecutamos el mismo programa y si no existe ningún argumento en la pila este mandara un **Try To Recovery Memory?** y en algunos casos ya no se podrá recuperar la memoria ocasionando la perdida de lo que tengamos grabado en el puerto 0 de la HP49G.

Para hacer un programa en SysRPL la sintaxis cambia es decir para iniciar un programa en User nosotros colocamos **«** *contenido del programa en user* **»** en cambio para hacer un programa en System nosotros tenemos que colocar lo siguiente:

```
::
contenido del programa en system
;
```

NOTA: *Todo esto dentro de comillas. en system se deben respetar las mayúsculas y minúsculas de lo contrario mandara error al momento de ensamblar el código de fuente.*

Existen varios programas para ensamblar como el ***JAZZ*** *que requiere también la librería* ***HPTABBS,*** *o también se puede ensamblar adjuntado la librería 256 y 257 que viene incorporada en la HP49G. escribiendo lo siguiente* ***{256 257} ATTACH*** *y teniendo instalada la librería* ***EXTABLE*** *que viene con la ROM de la HP de lo contrario mandara error al compilar Un excelente editor para hacer nuestros programas en System es el* ***EMACS*** *y de preferencia se deberá tener instalado con la librería* ***SDIAG,*** *la librería Emacs también te permite descompilar programas en System para poder ver su código de fuente y también tiene ayuda de algunos comandos de System. Algunos ensambladores requieren que el código termine con un arroba por ejemplo:*

```
::
contenido del programa en system
:
@
```

Una de las ventajas del system es que también se pueden usar comandos de USER anteponiendo una **x**,

Por ejemplo si queremos el comando **CHOOSE** en system seria **xCHOOSE.**
La **x** indica que se trata de un comando en USER.

2- MI PRIMER PROGRAMA EN SYSTEM

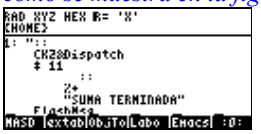
A continuación veremos un ejemplo paso a paso de cómo hacer nuestro primer programa en System el ejemplo se realizara usando el compilador interno de la HP49 para esto tenemos que escribir lo siguiente **{256 257} ATTACH** y presionamos **ENTER.**

También debemos tener instalada la librería **EXTABLE** que viene con la ROM de la HP49. Para poder compilar tendremos que tener la casilla **92** activada como se muestra en la figura esto se activa entrando a **MODE** y después a **FLAGS.**

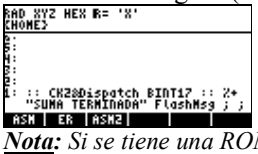
Copiamos el siguiente programa en la HP49G, respetando mayúsculas y minúsculas. Para empezar a escribir primero debemos abrir comillas **“ ”** y después copiar el texto que se muestra a continuación.

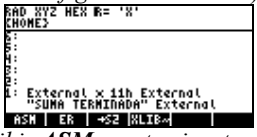
```
::
CK2&Dispatch
# 11
   ::
     %+
      "SUMA TERMINADA"
      FlashMsg
   ;
;
@
```

Nota: *Al escribir un programa en system primero deberemos abrir comillas y después escribir el programa como se muestra en la figura.*



Después nos vamos a librerías y debe figurar **MASD.** (ROM 1.18) *(Ver introducción para ver como adjuntar esta librería)* Como se muestra en la figura de arriba y presionamos **F1** (MASD) y deberá aparecer lo que se muestra en la figura (*Si se tiene desactivada la casilla **85** aparecerá como en la fig. de la derecha.*).





Nota: *Si se tiene una ROM superior a la 1.18 como la 1.19-6 Se deberá escribir **ASM.** y esto ejecutara **MASD**.*

Después se deberá grabar con un nombre por ejemplo ‘**SUMA**’ y presionamos **STO** y listo tenemos nuestro primer programa en SystemRPL.

A continuación veremos para que sirve cada comando de este ejemplo:

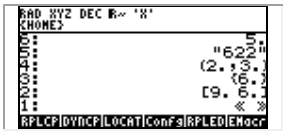
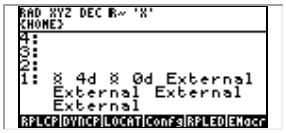
```
::                          Inicia System
CK2&Dispatch                Verifica que existan dos argumentos y despacha el tipo de objetos.
# 11                        Indica que el objeto del nivel 1 y 2  sean números reales.
   ::                       Inicia una subrrutina
   %+                       Suma dos números reales.
   "SUMA TERMINADA"         Escribe el mensaje SUMA TERMINADA
   FlashMsg                 Muestra el mensaje
   ;                        Termina la subrrutina
;                           Termina el System
@                           código para ensamblar con el MASD ( Si se usa JAZZ no es necesario el
                            arroba y antes de colocar las comillas  se deberá colocar el signo $ )
```

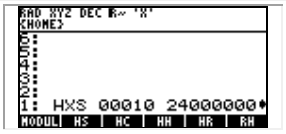
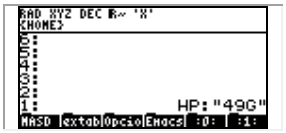
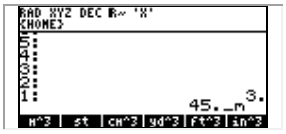
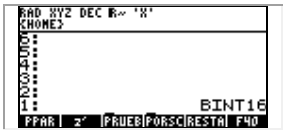
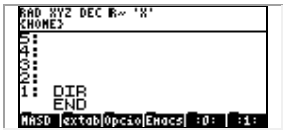
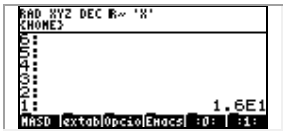
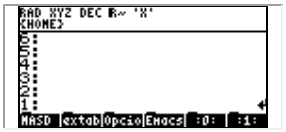
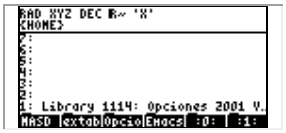
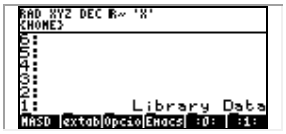
3- ALGUNAS IGUALDADES DE LOS COMANDOS DE USER

A continuación se verán algunas igualdades de los comandos usados en User y su equivalente en System:

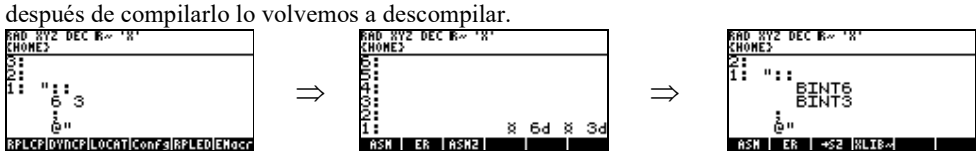
ERASE = DOERASE
SWAP = SWAP
➔ STR = DO>STR
SCROLL = ViewObject
SAME = EQUAL
DUP = DUP
VARS = DOVARS
PATH = PATHDIR
UPDIR = UPDIR
HOME = HOMEDIR
CRDIR = CREATEDIR
PGDIR = XEQPGDIR
ORDER = XEQORDER
STO = ?STO_HERE
DEPTH = DEPTH
DUP = DUP
DUP = dup
NDUPN = NDUPN
DROP = DROP
LCD➔ = DOLCD>
LCD➔ = HARDBUFF
LIST➔ = INNERCOMP

4- TIPOS DE OBJETOS

DESCRIPCION	TIPO			EJEMPLO	SIGNO
		(Entero Binario)			
Cualquier Objeto	any	0			
Número Real	real	1			%
Número Complejo	cmp	2			C%
Cadena de Caracteres (String)	str	3			$
Formación Vector o Matriz	arry	4			
Lista	list	5			
Nombre Global (ID)	idnt	6			
Nombre Local (Variable Temporal)	lam	7			
Programa o Secundario	seco	8			
Operación Algebraica	symb	9			
Clase Simbólico Cualquiera de los Tipos: 6 , 7, 9.	sym	10	#A		

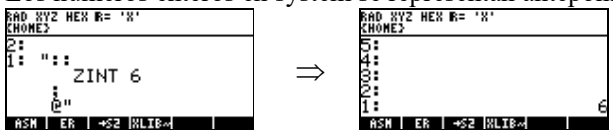
Cadena HXS o Número Binario	hxs	11	#B		#
Objeto Grafico	grob	12	#C		
Objeto Etiquetado	TAGGED	13	#D		
Objeto Unidad	unitob	14	#E		
Puntero ROM (Nombre XLIB)	rompointer	15	#0F		
Entero Binario		31	#1F		
Directorio		47	#2F		
Real Extendido		63	#3F		%%
Carácter (CHR)		111	#6F		
Objeto Código		127	#7F		
Librería			#8F		
Backup			#9F		
Datos de Librería			#AF		

Al escribir un programa en System el número Entero Binario se puede representar escribiendo solamente el número y al compilarlo este se convertirá en Entero Binario, esto se puede ver en el siguiente ejemplo que después de compilarlo lo volvemos a descompilar.

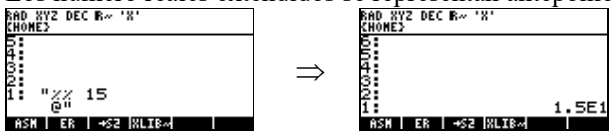


Los números reales en system se representan anteponiendo el signo **%**.

Los números enteros en system se representan anteponiendo la palabra **ZINT.**

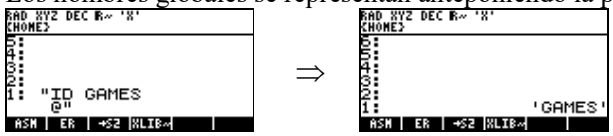


Los número reales extendidos se representan anteponiendo el signo **%%**.

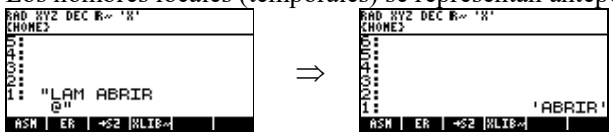


Los número complejos se representan anteponiendo el signo **C%**.

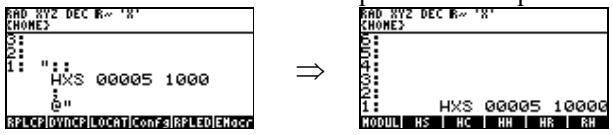
Los nombres globales se representan anteponiendo la palabra **ID**.



Los nombres locales (temporales) se representan anteponiendo la palabra **LAM**.



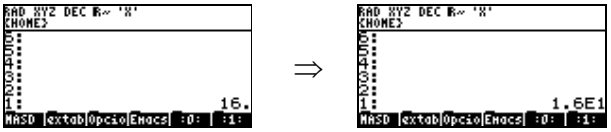
Las cadenas hexadecimaes se representan anteponiendo la palabra **HXS**.



5 CONVERSIÓN DE OBJETOS

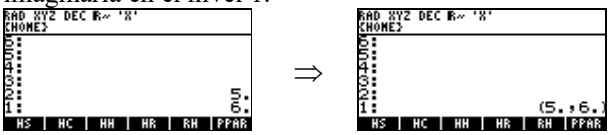
2FFAC %>%%

Convierte un número real a un número real extendido.



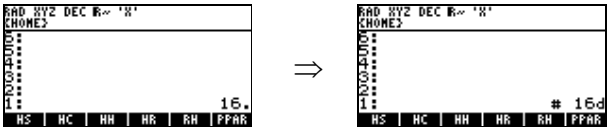
05C27 %>C%

Convierte un número real a un número complejo, requiere la parte real en el nivel 2 y la parte imaginaria en el nivel 1.



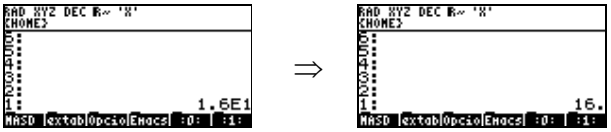
2EFCB %>#

Convierte un número real a una cadena HXS.



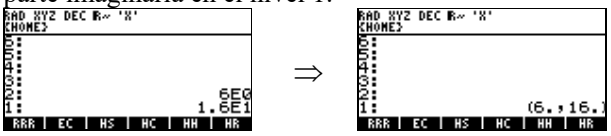
2FF9B %%>%

Convierte un número real extendido a un número real.



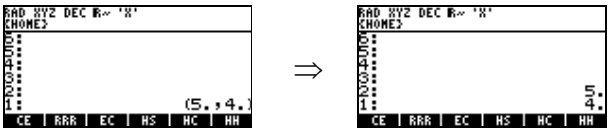
261CF %%>C%

Convierte un número real extendido a un número complejo, requiere la parte real en el nivel 2 y la parte imaginaria en el nivel 1.



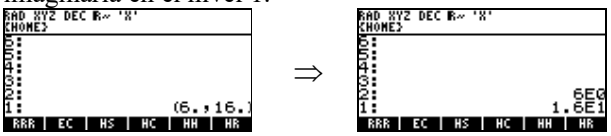
05D2C C%>%

Convierte un complejo a un número real, devuelve la parte real en el nivel 2 y la parte imaginaria en el nivel 1.



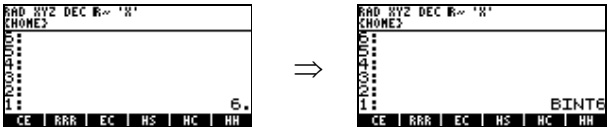
25E82 C%>%%

Convierte un complejo a un número real extendido, devuelve la parte real en el nivel 2 y la parte imaginaria en el nivel 1.



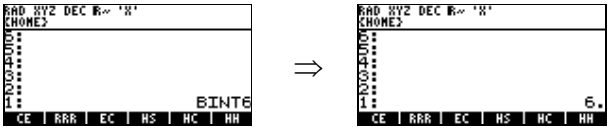
262F1 COERCE

Convierte un número real a un entero binario.



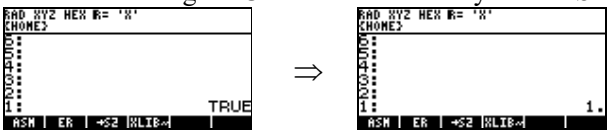
262F6 UNCOERCE

Convierte un entero binario a un número real.



2602B COERCEFLAG

Convierte un flag **TRUE** al numero real **1** y el **FALSE** al numero real **0**.



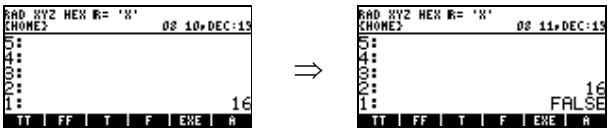
Los siguientes comandos sirven para preguntar el tipo de objeto que se encuentra en el nivel 1 de la pila devolviendo **TRUE** (Cierto) si el objetos que preguntamos se encuentra en el nivel 1 de la pila o **FALSE** (Falso), si no se encuentra.

TYPEBINT?	*Pregunta si es un entero binario*
TYPECARRY?	*Pregunta si es una formación compleja.*
TYPECHAR?	*Pregunta si es un carácter.*
TYPECMP?	*Pregunta si es un número complejo.*
TYPECOL?	*Pregunta si es un programa.*
TYPECSTR?	*Pregunta si es una cadena.*
TYPEEXT?	*Pregunta si es una unidad.*
TYPEGROB?	*Pregunta si es un grob.*
TYPEHSTR?	*Pregunta si es una cadena HEX.*
TYPEIDNT?	*Pregunta si es un nombre global.*
TYPELAM?	*Pregunta si es un nombre local (variable temporal)*
TYPELIST?	*Pregunta si es una lista.*
TYPEROM?	*Pregunta si es un puntero ROM (nombre XLIB).*
TYPERRP? .	*Pregunta si es un directorio*
TYPESYMB?	*Pregunta si es un objeto simbólico.*
TYPETAGGED?	*Pregunta si es un objeto etiquetado*

.***NOTA:*** *Anteponiendo la palabra* **DUP** *el objeto se duplica.*

Ejemplo:

```
::
 DUPTYPEIDNT?
;
@
```



Este ejemplo duplica el objeto del nivel 1 y pregunta si es un nombre global. Si el objeto es un nombre global devolverá la bandera **TRUE.**

6- ALGUNOS TIPOS DE OBJETOS

OBJETO ENTERO BINARIO
El uso de este tipo de objetos es para representar enteros binarios cuya precisión es equivalente a una dirección de memoria.

OBJETO CADENA HEXADECIMAL (HEX)
Un uso típico de este tipo de objetos es un búfer o tabla.

OBJETO CARACTER
Este objeto se usa para representar cantidades de un solo byte.

OBJETO CODIGO
Un objeto código es una sección en lenguaje ensamblador.

Ejemplo:

```
::
 CODE
  Programa en lenguaje ensamblador
 ENDCODE
;
@
```

OBJETO SIMBOLICO
Este tipo de objetos se usa para representar objetos simbólicos para aplicaciones de matemática simbólica.

7- FUNCIONES DE COMPARACIÓN O TEST

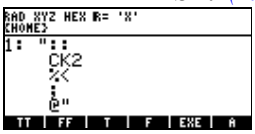
Las funciones de comparación son aquellas que comparan argumentos, devolviendo **TRUE** si es verdadero o **FALSE** si es falso de acuerdo a lo que se este preguntando.

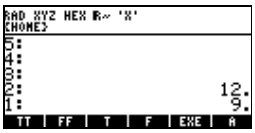
Estas funciones se describen en la siguiente Tabla:

COMANDO	DESCRIPCIÓN
EQ	Pregunta si dos objetos ocupan físicamente el mismo espacio en la memoria.
EQUAL	Pregunta si dos objetos son iguales
=	Pregunta si dos objetos son iguales
<=	Pregunta si el objeto 2 es menor o igual al objeto 1
<	Pregunta si el objeto 2 es menor al objeto 1
>	Pregunta si el objeto 2 es mayor al objeto 1
<=	Pregunta si el objeto 2 es menor o igual al objeto 1
>=	Pregunta si el objeto 2 es mayor o igual al objeto 1
<>	Pregunta si el objeto 2 es diferente al objeto 1

Nota: *Al comparar el tipo de objetos se deberá preguntar que tipo de objeto es, por ejemplo si se trata de un número real se deberá anteponer el Signo %.*

Ejemplo:
En este ejemplo se deberán introducir dos números reales en la pila y devolverá **TRUE** si el objeto 2 es menor al objeto 1. Por ejemplo si introducimos **12.** y **9.** y ejecutamos el siguiente programa devolverá **FALSE.** *(Para los siguientes ejemplos se puede activar el flag 85 para ver la pila en modo system)*





⇒

8- FUNCIONES LÓGICAS

Las funciones Lógicas son aquellas que permiten dar a conocer la relación entre dos condiciones.
Estas funciones son el **AND** y el **OR**.

03B46 AND

Esta función permite dar a conocer si se cumplen Simultáneamente dos condiciones.

2: Flag 1 **1:** Flag 2	➔	**1:** Flag

Si flag 1 y flag 2 son ambos **TRUE** entonces el flag que se devuelve es **TRUE** si no, **FALSE.**
Esto se muestra en la siguiente tabla:

CONDICIÓN 1	CONDICIÓN 2	RESPUESTA
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

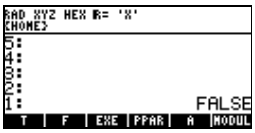
03B75 OR

Si el flag 1 o el flag 2 (o ambos) son **TRUE** entonces **TRUE** de lo contrario **FALSE**.
Con esta función se puede conocer si una de las dos condiciones se cumple.
Esto se muestra en la siguiente tabla:

CONDICIÓN 1	CONDICIÓN 2	RESPUESTA
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

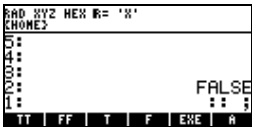
8.1 - OTROS COMANDOS DE TESTS

03AC0 FALSE
Pone la bandera **FALSE** en el nivel 1 de la pila

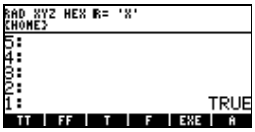


36554 FALSETRUE
Pone la bandera **FALSE** en el nivel 2 y **TRUE** en el nivel 1 de la pila.

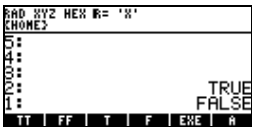
369FF FALSE'
Pone la bandera **FALSE** en el nivel 2 y **:: ;** en el nivel 1 de la pila.



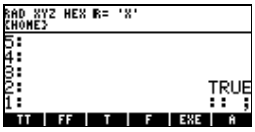
03A81 TRUE
Pone la bandera **TRUE** en el nivel 1 de la pila.



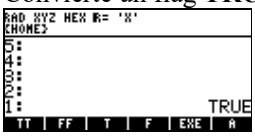
36540 TRUEFALSE
Pone la bandera **TRUE** en el nivel 2 y **FALSE** en el nivel 1 de la pila.



369D2 TRUE'
Pone la bandera **TRUE** en el nivel 2 y **:: ;** en el nivel 1 de la pila.



03AF2 NOT
Convierte un flag **TRUE** en un **FALSE**. y un **FALSE** en un **TRUE**.



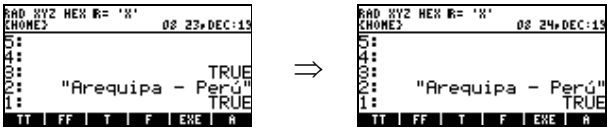
⇔

35C7C NOTAND

Devuelve **TRUE** si se encuentra **TRUE** y **FALSE** de lo contrario devuelve **FALSE**.

35CB8 ROTAND

Requiere un flag 1 un objeto y un flag 2, después hace un **ROT** y luego un Y lógico. (**AND**). El ejemplo que se muestra abajo es el único caso donde devuelve **TRUE**, en los demás casos devolverá **FALSE**.



03ADA XOR

Si el flag 1 y flag 2 son **TRUE** o **FALSE** entonces devuelve **FALSE**, si no devuelve **TRUE**. Esto se muestra en la siguiente tabla:

CONDICIÓN 1	CONDICIÓN 2	RESPUESTA
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	FALSE

2602B COERCEFLAG

Convierte un flag **TRUE** en el número real **1.**

Convierte un flag **FALSE** en el número real **0**.

9 - VERIFICAR EL TIPO DE ARGUMENTO

Para verificar el tipo de argumento que se tiene en la pila se utilizan los siguientes comandos. **CK&DISPATCH0** y **CK&DISPATCH1** seguido del tipo de objeto y a continuación el procedimiento.

También se puede verificar con el siguiente comando. **CKN&Dispatch,** siendo **N** el número de argumentos que se necesitan en la pila por ejemplo si se necesitan dos argumentos se pondrá **CK2&Dispatch.**

Ejemplo:

```
::
CK1&Dispatch
idnt
PURGE
;
@
```

Este ejemplo verifica primero que exista un argumento en la pila y luego verifica si lo que se tiene en la pila es un nombre global, de ser así se procederá a borrar el nombre local.

9.1 NUMERO DE ARGUMENTOS

Las siguientes palabras verifican que existan de 0-5 argumentos en la pila y pemiten el mensaje de error **"Too Feb Arguments"** si no es así.

CK0, CK0NOLASTWD	No se requiere ningún argumento
CK1, CK1NOLASTWD	Se requiere un argumento
CK2, CK2NOLASTWD	Se requieren dos argumentos
CK3, CK3NOLASTWD	Se requieren tres argumentos
CK4, CK4NOLASTWD	Se requieren cuatro argumentos
CK5, CK5NOLASTWD	Se requieren cinco argumentos

262CE CKN
Verifica n argumentos en la pila.

Si queremos asegurarnos que en cada nivel de la pila exista el tipo de objeto que nosotros necesitamos se puede especificar de la siguiente manera:

#nnnnn

Cada n representa un tipo de objeto diferente.

Ejemplo 1:
Queremos verificar que en el nivel 2 de la pila este un grafico y en el nivel uno un número real.

```
::
 CK2&Dispatch
 #C1
;
@
```

En este ejemplo la **C** indica un grafico el **1** un número real. *(Ver tipos de objetos)*

Ejemplo 2:

Queremos verificar que en el nivel 3 de la pila este un directorio, en el nivel dos una librería y el nivel 1 un número real

::

```
CK3&Dispatch
#2F8F1
```

;

```
@
```

En este ejemplo la **2F** indica un directorio, **8F** una librería y el 1 un número real. *(Ver tipos de objetos)*

10 - INDICADORES DE SISTEMA (Flags)

25F23 SaveSysFlags
Guarda los Flags de sistema.

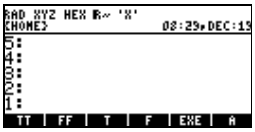
25F22 RestoreSysFlags
Restaura los Flags guardados de sistema.

26152 SetUserFlag
Activa el flag de usuario indicado.

2614D SetSysFlag
Activa el flag de sistema indicado.

Ejemplo:

```
::
40
SetSysFlag
;
@
```

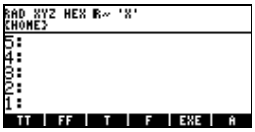


Este ejemplo activa el flag número 40 (Indicador del reloj)

26044 ClrSysFlag
Desactiva el flag de sistema indicado.

Ejemplo:

```
::
 40
 ClrSysFlag
;
@
```



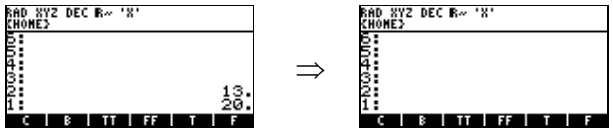
Este ejemplo desactiva el flag número 40 (Indicador del reloj)

26049 ClrUserFlag
Desactiva el flag de usuario indicado.

11. OPERACIONES DE LA PILA

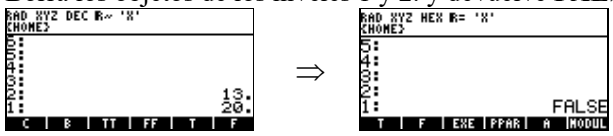
Las palabras listadas en este capítulo realizan operaciones de la pila sencillas o múltiples.

03258 2DROP
Borra los objetos de los niveles 1 y 2.

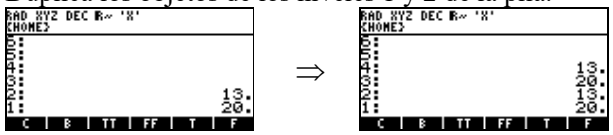


355A5 2DROP00
Borra los objetos de los niveles 1 y 2 y devuelve los números binarios 0 en el nivel 1 y 2 de la pila.

35B32 2DROPFALSE
Borra los objetos de los niveles 1 y 2. y devuelve **FALSE** en el nivel 1 de la pila.



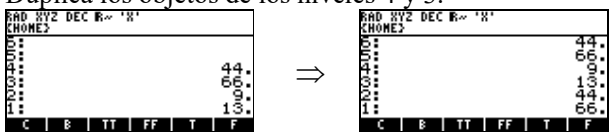
031AC 2DUP
Duplica los objetos de los niveles 1 y 2 de la pila.



36CA4 2DUP5ROLL
Duplica los objetos de los niveles 1 y 2 de la pila y trae al objeto del nivel 5 de la pila.

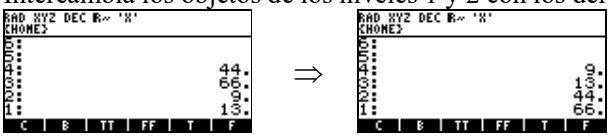
35D30 2DUPSWAP
Duplica los objetos de los niveles 1 y 2 de la pila y intercambia de niveles el 1 con el 2.

37046 2OVER
Duplica los objetos de los niveles 4 y 3.



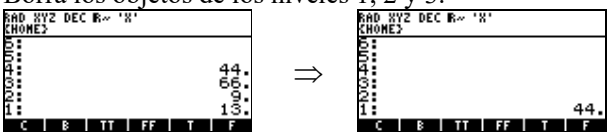
35018 2SWAP

Intercambia los objetos de los niveles 1 y 2 con los del 3 y 4.



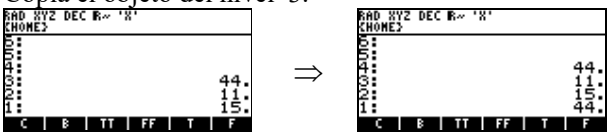
341D2 3DROP

Borra los objetos de los niveles 1, 2 y 3.



34485 3PICK

Copia el objeto del nivel 3.



3422B 3UNROLL

Rota un nivel hacia abajo los objetos de los niveles 1, 2 y 3.

341D7 4DROP

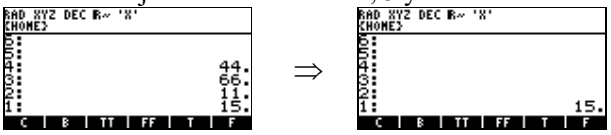
Borra los objetos de los niveles 1, 2 , 3 y 4.

34331 4UNROLL

Rota un nivel hacia abajo los objetos de los niveles 1, 2, 3 y 4.

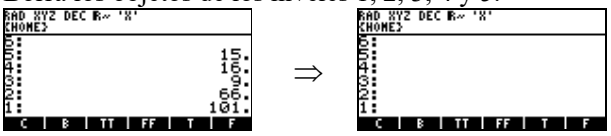
343CF 4UNROLL3DROP

Borra los objetos de los niveles 2, 3 y 4.

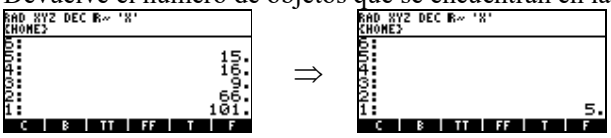


341DC 5DROP

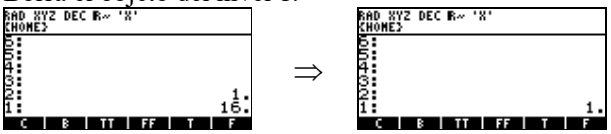
Borra los objetos de los niveles 1, 2, 3, 4 y 5.



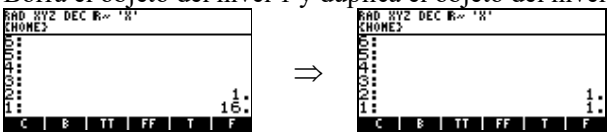
0314C DEPTH
Devuelve el número de objetos que se encuentran en la pila.



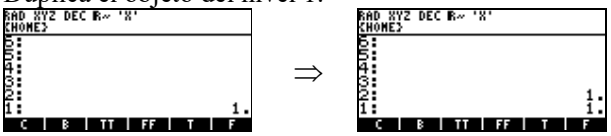
03244 DROP
Borra el objeto del nivel 1.



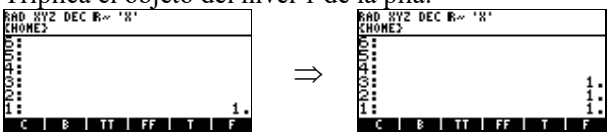
357CE DROPDUP
Borra el objeto del nivel 1 y duplica el objeto del nivel 2.



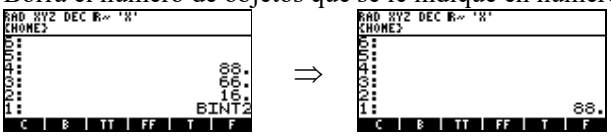
03188 DUP
Duplica el objeto del nivel 1.



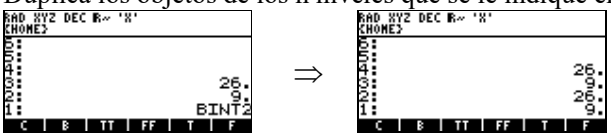
35CE0 DUPDUP
Triplica el objeto del nivel 1 de la pila.



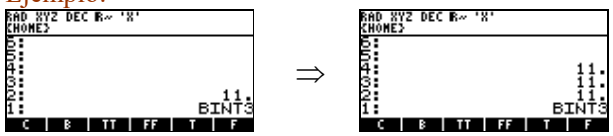
0326E NDROP
Borra el número de objetos que se le indique en números enteros binarios.



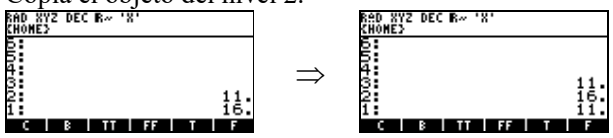
031D9 NDUP
Duplica los objetos de los n niveles que se le indique en números enteros binarios.



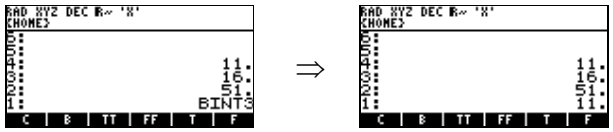
28143 NDUPN
Duplica el objeto del nivel 1 el número de veces que se le indique en números binarios devolviendo el número de copias que hizo en números binarios.
Ejemplo:



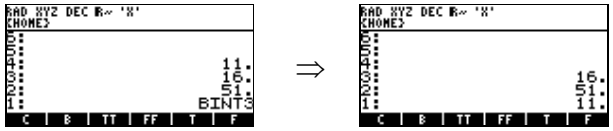
032C2 OVER
Copia el objeto del nivel 2.



032E2 PICK
Copia al nivel 1 cualquier objeto de los niveles este requiere el número del nivel a copiar en números binarios.



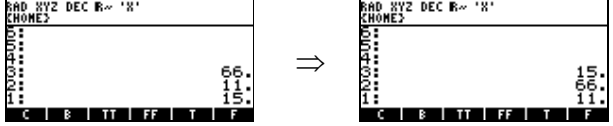
03325 ROLL
Cambia la posición de los objetos que se encuentran en n niveles de abajo hacia arriba, esto requiere el número del nivel en números binarios.



03295 ROT
Cambia la posición de los objetos de los niveles 1, 2 y 3, de abajo hacia arriba.

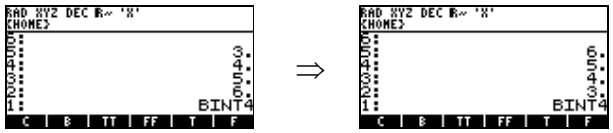
03223 SWAP
Intercambia los objetos de los niveles 1 y 2.

3422B UNROT
Cambia la posición de los objetos de los niveles 1, 2 y 3, de arriba hacia abajo.



28187 reversym

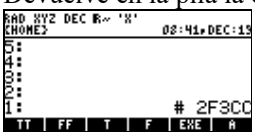
Invierte el orden de los objetos de la pila, esto requiere el número de objetos a invertir en números enteros binarios.



12 - UTILIDADES ADICIONALES DE MEMORIA

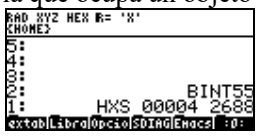
05F42 GARBAGE
Fuerza una recolección de basura.

05F61 MEM
Devuelve en la pila la cantidad de memoria libre en nibbles (no se fuerza una recolección de basura)

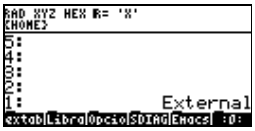


05944 OCRC
Devuelve la cantidad de memoria que ocupa un objeto en nibbless y el checksum en hxs.

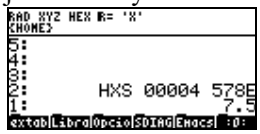
⇒



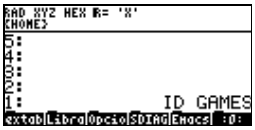
2F257 OCRC%
Devuelve el checksum de un objeto en hxs y la cantidad de memoria que ocupa el objeto en bytes



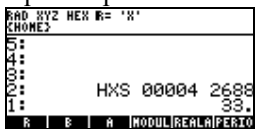
⇒



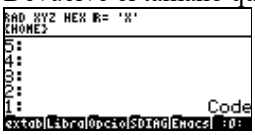
2F267 VARSIZE
Devuelve el checksum y tamaño que ocupa una variable en bytes.



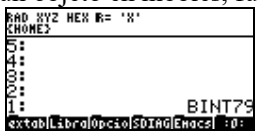
⇒



05902 OSIZE
Devuelve el tamaño que ocupa un objeto en nibbles, fuerza a una recolección de basura.



⇒



Si se desea saber cuanto de memoria libre existe en algún puerto existe el comando en UserRpl **PVARS** que en system seria **xPVARS** cuya dirección de memoria es **3EA49.** Este comando requiere el número del puerto en números reales y devuelve en el nivel 2 una lista con los objetos almacenados en el puerto y en el nivel 1 la cantidad de memoria libre en dicho puerto.

⇒

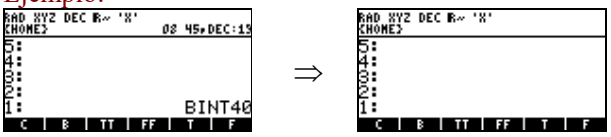
13 - COMANDOS DEL SISTEMA

Las siguientes palabras ponen, comprueban o controlan varias condiciones o modos del sistema.

25FA9 ALARM?
Devuelve **TRUE** si esta activada una alarma.

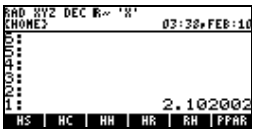
2EF6C AtUserStack
Declara propiedad del usuario todos los objetos de la pila.

26044 ClrSysFlag
Borra la bandera del sistema, se debe introducir la bandera a borrar en números binarios.
Ejemplo:

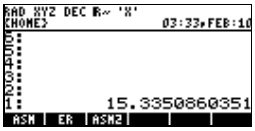


26049 ClrUserFlag
Borra la bandera de usuario, se debe introducir la bandera a borrar en números binarios.

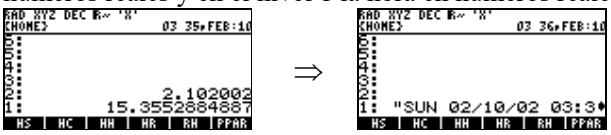
2EED0 DATE
Devuelve la fecha en números reales.



2EECF TOD
Devuelve la hora en números reales.



2EED3 TIMESTR
Devuelve en un string el día, la fecha y la hora, requiere como entrada en el nivel 2 la fecha en números reales y en el nivel 1 la hora en números reales.

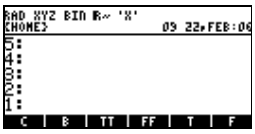


25EB2 DOBEEP
Emite un sonido mediante un frecuencia este requiere como entrada la frecuencia que esta dada en Hz en números reales y el tiempo de duración que esta dado en segundos en números reales.
Ejemplo:

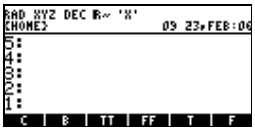
```
::
%1500
%2
DOBEEP
;
@
```

Este ejemplo generara un sonido de frecuencia de 1500 Hz con una duración de 2 segundos.

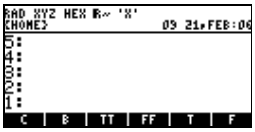
2EFA6 DOBIN
Pone la base en modo **BIN**ario.



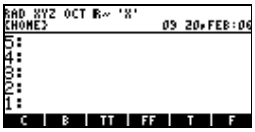
2EFA8 DODEC
Pone la base en modo **DEC**imal.



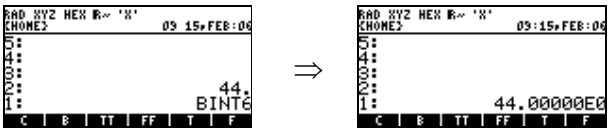
2EFA5 DOHEX
Pone la base en modo **HEX**adecimal.



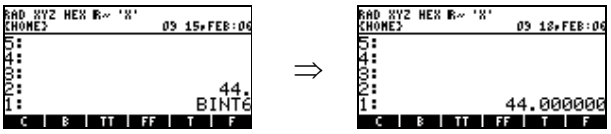
2EFA 7 DOOCT
Pone la base en modo **OCT**al.



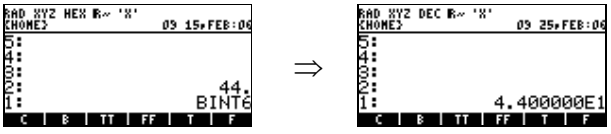
2604E DOENG
Pone la pantalla en modo ingenierias **ENG** desde (0-11) dígitos. se requiere el número de digitos en números binarios.



26053 DOFIX
Pone la pantalla en modo **FIX** desde (0-11) dígitos. Se requiere el número de digitos en números binarios.



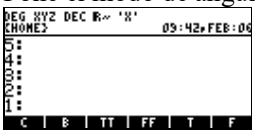
26058 DOSCI
Pone la pantalla en modo científico **SCI** con (0-11) dígitos se requiere el número de digitos en números binarios.



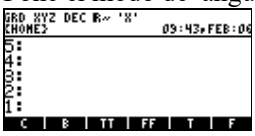
2605D DOSTD
Pone la pantalla en modo estándar **STD.**

25EBA DPRADIX?
Devuelve **TRUE** si la coma actual es un punto (**.**).
Devuelve **FALSE** si la coma actual es una coma (**,**).

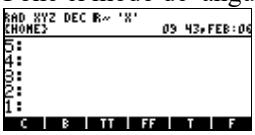
2FFBD SETDEG
Pone el modo de ángulo en **DEGREES** (SEXAGESIMALES)



2FFEF SETGRAD
Pone el modo de ángulo en **GRADS** (CENTESIMALES)

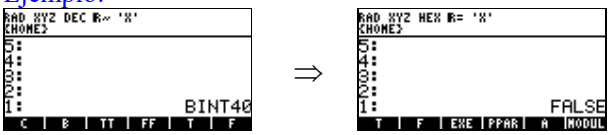


2FFDB SETRAD
Pone el modo de ángulo en **RADIANS** (RADIANES)



26120 SLOW
Retardo de 15 milisegundos

26170 TestSysFlag
Devuelve **TRUE** si una bandera especifica del sistema esta activada, requiere como entrada el número de la bandera de sistema en números binarios.
Ejemplo:



Esto escrito en un programa seria de la forma siguiente.

```
::
40
TestSysFlag
;
@
```

Devuelve **TRUE** si el indicador del reloj esta activado (flag 40), y **FALSE** si esta desactivado.

26175 TestUserFlag
Devuelve **TRUE** si una bandera de usuario esta activada, requiere como entrada un número binario.

26125 VERYSLOW
Retardo de 300 milisegundos

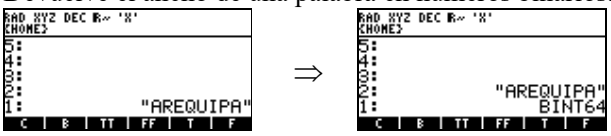
2612A VERYVERYSLOW
Retardo de 3 segundos.

25F95 LANGUAGE>
Devuelve el número entero binario del lenguaje actual.
0 = Ingles.
1 = Francés.
2 = Español.

25F90 >LANGUAGE
Cambia los mensajes de error al lenguaje indicado se requiere el número entero binario 0, 1 o 2.
Ejemplo:

```
::
2
>LANGUAGE
;
@
```

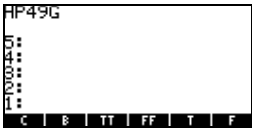
2EFBE WORDSIZE
Devuelve el ancho de una palabra en números binarios.



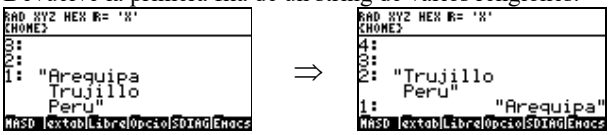
2EFAA dostws
Almacena el ancho de una palabra, requiere como entrada el ancho de la palabra en números binarios.

2F2D4 dowait
Hace una pausa durante un número de segundos en estado de sueño ligero. Requiere el tiempo de espera en números reales.
Ejemplo:
*Este ejemplo muestra el mensaje **HP49G** durante 2 segundos.*

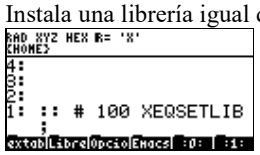
```
::
"HP49G"
FlashMsg
%2
dowait
;
@
```



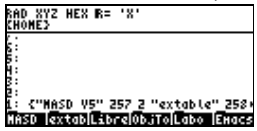
25EF8 SEP$NL
Devuelve la primera fila de un string de varios renglones.



2F2A7 XEQSETLIB
Instala una librería igual que el comando de User **ATTACH**. Por ejemplo para instalar la librería 256.

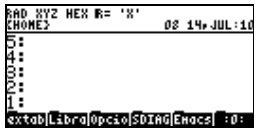


FPTR 2 14
Devuelve el Nombre, ID y puerto de las librerias instaladas.

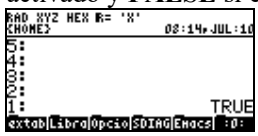


2F21E `ViewStrObject`
Necesita un flag y un string.
Si flag es **TRUE** visualiza el string como un texto.
Si flag es **FALSE** visualiza el string como un grob.

048F9 DispTimeReq2?
Devuelve **TRUE** si el reloj esta activado y **FALSE** si esta desactivado.



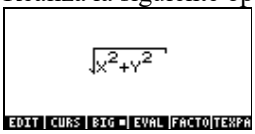
⇒



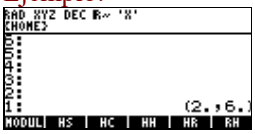
13 - FUNCIONES DE NUMEROS COMPLEJOS

25E81 C%1/ (C% ➔ C%')
Inverso

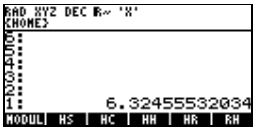
25E84 C%ABS
Realiza la siguiente operación del número complejo (x , y).



Los eleva al cuadrado y les saca la raíz.
Ejemplo:



⇒



25E85 C%ACOS (C% ➔ C%')
Arco-coseno

25E87 C%ALOG (C% ➔ C%')
Antilogaritmo base 10.

25E88 C%ARG (C% ➔ %)
Devuelve ANGULO(x,y) de (x,y)

25E89 C%ASIN (C% ➔ C%')
Arco-seno

25E8B C%ATAN (C% ➔ C%')
Arco-tangente

25E8F C%C^C (C%1 C%2 ➔ C%3)
Potencia

261ED C%CHS (C% ➔ C%')
Cambio de signo

261DE C%%CHS (C%% ➔ C%%')
Cambio de signo

261F2 C%CONJ (C% ➔ C%')
Conjugado

261E3 C%%CONJ (C%% ➔ C%%')
Conjugado

25E8D C%COS (C% ➔ C%')
Coseno

25E8E C%COSH (C% ➔ C%')
Coseno hiperbólico

25E92 C%LN (C% ➔ C%')
Logaritmo natural

25E93 C%LOG (C% ➔ C%')
Logaritmo base 10

25E95 C%SGN
Realiza la siguiente operación con el número complejo (x , y).
Con **X** realiza lo que se muestra en la Figura Nº1 y con **Y** lo que se muestra en la Figura Nº2.

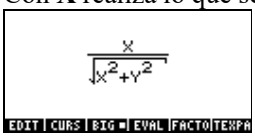


Figura Nº1

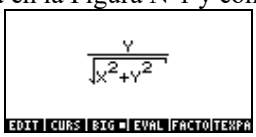
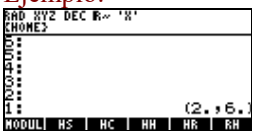
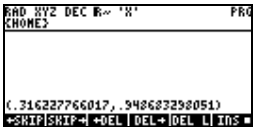


Figura Nº2

Ejemplo:



⇒


25E96 C%SIN (C% ➔ C%')
Seno

25E97 C%SINH (C% ➔ C%')
Seno hiperbólico

25E98 C%SQRT (C% ➔ C%')
Raiz cuadrada

25E99 C%TAN (C% ➔ C%')
Tangente

25E9A C%TANH (C% ➔ C%')
Tangente hiperbólica

14 - NUMEROS REALES

Los números reales se escriben con % y los números reales extendidos se escriben con %%.
A continuación se muestran algunas funciones de estos números.

14.1 - FUNCIONES DE NUMEROS REALES

30385 %%*
Multiplica dos números reales extendidos.

3032E %%+
Suma dos números reales extendidos.

3033A %%-
Resta dos números reales extendidos.

302DB %%ABS
Valor absoluto de un número real extendido.

306F3 %%ACOSRAD
Arco-coseno de un número real extendido, usando radianes.

306C3 %%ASINRAD
Arco-seno de un número real extendido, usando radianes.

302FB %%CHS
Cambia de signo a un número real extendido.

30642 %%COS
Coseno de un número real extendido.

30653 %%COSDEG
Coseno con grados sexagesimales de un número real extendido.

307B2 %%COSH
Coseno hiperbólico de un número real extendido.

30663 %%COSRAD
Coseno de un número real extendido, en radianes.

300C7 %%MAX
Devuelve el mayor de dos números reales extendidos

305F1 %%SIN
Seno de un número real extendido.

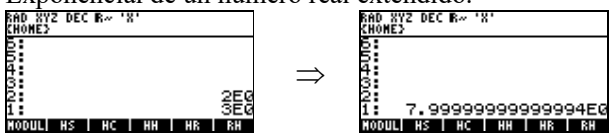
30602 %%SINDEG
Seno de un número real extendido, con grados sexagesimales.

30780 %%SINH
Seno hiperbólico de un número real extendido.

304D5 %%SQRT
Raiz cuadrada de un número real extendido.

30693 %%TANRAD
Tangente de un número real extendido, en radianes.

3044A %%^
Exponencial de un número real extendido.



3035F %+
Suma dos números reales.

25E69 %+SWAP
Suma dos números reales y luego hace SWAP. Requiere un objeto en el nivel 2 y el número real en el nivel 1 de la pila.

3036C %-
Resta dos números reales.

26F36 %1+
Suma uno a un número real.

26F4A %1-
Resta uno a un número real.

302EB %ABS
Valor absoluto de un número real.

306DC %ACOS
Arco-coseno de un número real.

307FE %ACOSH
Arco-coseno hiperbólico de un número real.

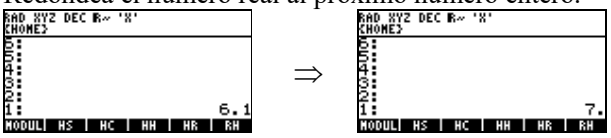
306AC %ASIN
Arco-seno de un número real.

307EB %ASINH
Arco-seno hiperbólico de un número real.

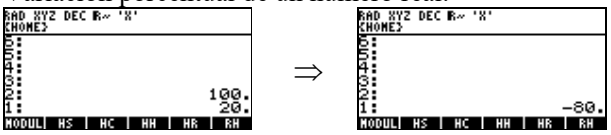
3070C %ATAN
Arco-tangente de un número real.

30811 %ATANH
Arco-tangente hiperbólico de un número real.

3095E %CEIL
Redondea el número real al próximo número entero.



3041B %CH
Variación porcentual de un número real.



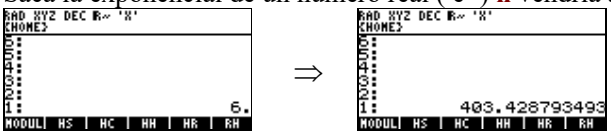
3030B %CHS
Cambia de signo al número real.

3062B %COS
Coseno de un número real.

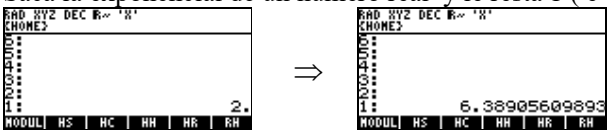
307C5 %COSH
Coseno hiperbólico de un número real.

3000D %D>R
Convierte un número real que este en grados sexagesimales a radianes

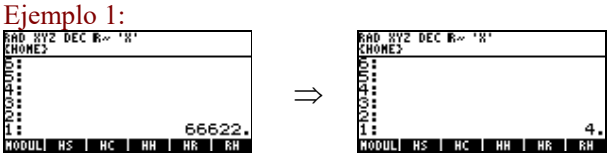
3051A %EXP
Saca la exponencial de un número real (e^n) **n** vendría a ser el número real.



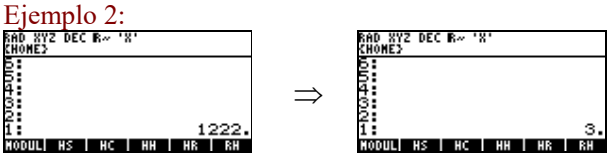
3052D %EXPM1
Saca la exponencial de un número real y le resta 1 (e^n-1) **n** vendría a ser el número real.



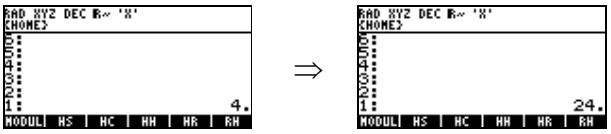
30824 %EXPONENT
Devuelve el exponente de un número real.
Ejemplo 1:



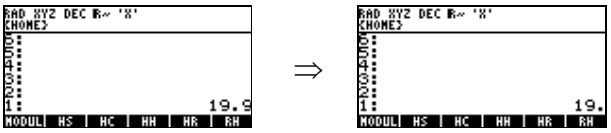
Ejemplo 2:



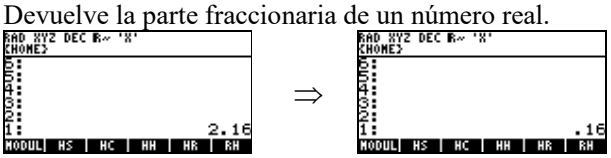
30AAF %FACT
Factorial de un número real.



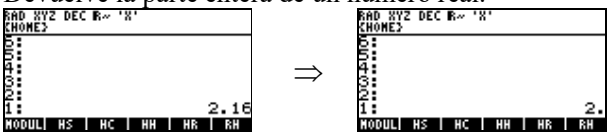
30971 %FLOOR
Redondea un número real al menor entero.



30938 %FP
Devuelve la parte fraccionaria de un número real.



3094B %IP
Devuelve la parte entera de un número real.



30559 %LN
ln(n), n es un número real.

30592 %LNP1
ln(n+1), n es un número real.

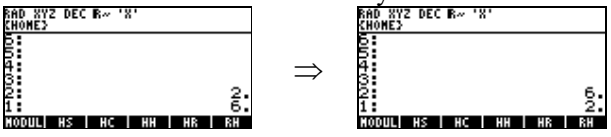
3056C %LOG
Logaritmo base 10 de un número real.

3031B %MANTISSA
Devuelve la mantisa de un número real.

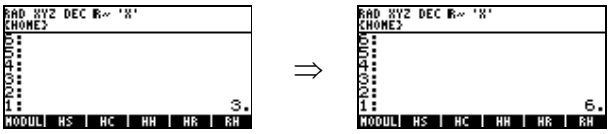
300E0 %MAX
Devuelve el mayor de dos números reales.

300F9 %MIN
Devuelve el menor de dos números reales.

35DBC %MAXorder (%1 %2 --> %mayor %menor)
Ordena dos números reales de mayor a menor.

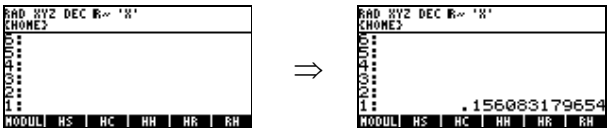


30837 %NFACT
Factorial de un número real.

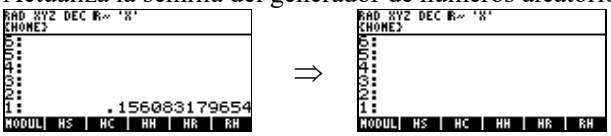


30040 %R>D
Convierte un número real que este en radianes a uno en grados sexagesimales

309AD %RAN
Genera un número real aleatorio



30A66 DORANDOMIZE
Actualiza la semilla del generador de números aleatorios



305DA %SIN
Seno de un número real.

30799 %SINH
Seno hiperbólico de un número real.

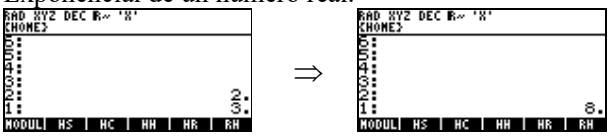
304F4 %SQRT
Raíz cuadrada de un número real.

303F6 %T
Porcentaje total de un número real.

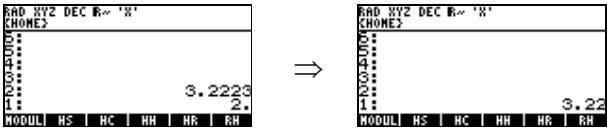
3067C %TAN
Tangente de un número real.

307D8 %TANH
Tangente hiperbólica de un número real.

3045B %^
Exponencial de un número real.

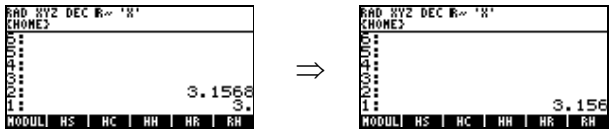


30F14 RNDXY
Redondea un número real a un determinado número de decimales.



30F28 TRCXY

Trunca un número real a determinados decimales.



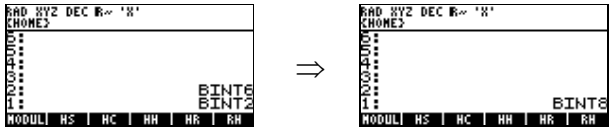
15 - MANIPULACION DE ENTEROS BINARIOS

03EC2 #*

Multiplica dos enteros binarios.

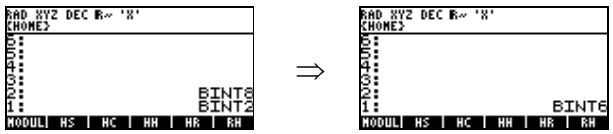
03DBC #+

Suma dos enteros binarios.



03DE0 #-

Resta dos enteros binarios.

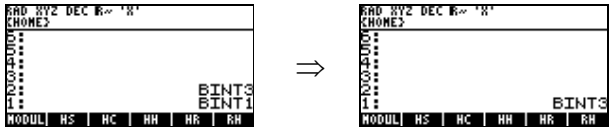


03EF7 #/

Divide dos enteros binarios, devuelve el residuo en el nivel 2, y el cociente en el nivel 1.

3551D #MAX

Devuelve el máximo de dos enteros binarios



16- ENTRADA DE DATOS

A continuación se verán algunos ejemplos para la entrada de datos en los programas. En algunos casos se vera primero el ejemplo en User y después el mismo ejemplo en System.

Ejemplo 1:

```
USER-RPL
« "TITULO" {{"SUBTITULO 1"} {"SUBTITULO 2"} {"SUBTITULO 1"}}
1 CHOOSE »
```

```
SYS-RPL
::
{"SUBTITULO 1"}
{"SUBTITULO 2"}
{"SUBTITULO 3"}
3
"TITULO"
0
FPTR 2 6F
;
@
```

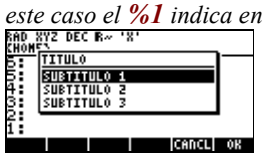
NOTA: *En el ejemplo en System el número entero binario **3** indica la cantidad de subtitulos que existen en este caso son 3 subtitulos y el número entero binario **0** indica en donde va a empezar la marcación del subtitulo en este caso el **0** indica en el primer subtitulo,*

Ejemplo 2:

```
USER-RPL
« "TITULO" {{"SUBTITULO 1"} {"SUBTITULO 2"} {"SUBTITULO 1"}}
1 CHOOSE »
```

```
SYS-RPL
::
"TITULO"
{
{"SUBTITULO 1"}
{"SUBTITULO 2"}
{"SUBTITULO 3"}
}
%1
FPTR 2 2D
;
@
```

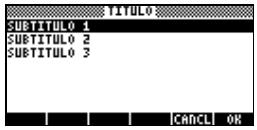
NOTA: *En el ejemplo en System el número real **%1** indica donde va a empezar la marcación del subtitulo en este caso el **%1** indica en el primer subtitulo,*



Ejemplo 3:

```
SYS-RPL
::
'
 ::
  60
  #=casedrop
  TrueTrue
 ;
"TITULO"
17
{
{"SUBTITULO 1"}
{"SUBTITULO 2"}
{"SUBTITULO 3"}
}
1
ROMPTR B3 0
;
@
```

NOTA: *En este ejemplo el número entero binario* ***1*** *indica donde va a empezar la marcación del subtitulo en este caso el* ***1*** *indica en el primer subtitulo.*

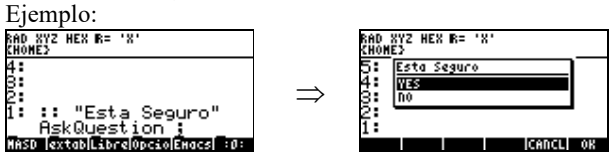


En estos 3 ejemplos si se sale pulsando **OK** devolverá el flag **TRUE**, si se sale pulsando **CANCL** devolverá el flag **FALSE.**

2F1A5 AskQuestion

Crea un CHOOSE con un string si se sale con **OK** devuelve el flag **TRUE** y si se sale con **CANCL** devuelve el flag **FALSE**.

Ejemplo:



17 – InputLine

2EF5F InputLine
Este comando requiere de 10 argumentos para funcionar, devuelve el flag **TRUE** si se salio con **ENTER** y devuelve el flag **FALSE** si se cancelo con **ON**, tiene la siguiente sintaxis.

SINTAXIS:
::
"Mensaje"
"Valor Inicial"
Posición del Cursor
Modo de escritura
Modo de entrada
Modo de Alpha
{Menús de edición }
Menú inicial del menú de edición
FALSE o TRUE
Proceso de edición
InputLine
;
@

Mensaje
Deberá ir en un string el mensaje que deseamos que aparezca.

Valor inicial
Deberá ir en un string el valor inicial que queremos que aparezca, si no queremos ninguno colocamos **NULL$**.

Posición del Cursor
Indica la posición inicial donde se quiere que este ubicado el cursor, si colocamos el número entero binario **#0** este va a indicar el final de la línea de edición

Modo de Escritura
Se debe colocar cualquiera de los siguientes números enteros binarios, cada uno indica un modo distinto.
0 = Modo escritura/sobreescritura actual
1 = Modo escritura normal
2 = Modo sobreescritura

Modo de Entrada
Se debe colocar cualquiera de los siguientes números enteros binarios, cada uno indica un modo distinto.

0 = Modo entrada actual mas modo programa
1 = Modo entrada inmediata
2 = Modo Algebraico

Modo de Alpha
Se debe colocar cualquiera de los siguientes números enteros binarios, cada uno indica un modo distinto.
0 = Modo alfa actual
1 = Modo alfa activado
2 = Modo alfa desactivado

Menús de Edición
Se debe colocar los menús que se desea que aparezcan, estos menús deberán estar dentro de una lista encerrados cada menú en un string. Si no se desea ningún menú colocamos **NULL{ }**.

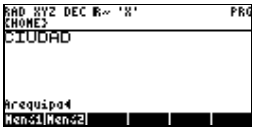
Menú inicial del menú de edición
Generalmente es **ONE** (uno), esto indica a partir de que menú se desea mostrar.

FALSE O TRUE
Si colocamos **TRUE** este especifica que si pulsamos **ON** mientras exista una línea de edición no nula, este abortaría el InputLine. Y si colocamos **FALSE** este simplemente borraría la línea de edición.

Proceso de Edición
Este indica como procesar la línea de edición resultante, se debe colocar cualquiera de los siguientes números enteros binarios, cada uno indica un proceso distinto.
0 = Devuelve la línea de edición en un string.
1 = Devuelve la línea de edición en un string y como un objeto analizado.
2 = Analiza y evalúa la línea de edición.

Ejemplo 1 :

```
::
“CIUDAD”
“Arequipa”
0
0
1
1
{“Menú1” “Menú2” }
1
FALSE
0
InputLine
;
@
```

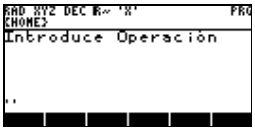


Este código se puede reducir a:

```
::
"CIUDAD"
"Arequipa"
ZEROZERO
ONEONE
{"Menú1" "Menú2" }
1
FALSE
0
InputLine
;
@
```

Ejemplo 2 :

```
::
"Introduce Operación"
"''"
2 0
1 2
NULL{}
ONE
FALSE
ONE
InputLine
;
@
```



18 - COMANDOS DE PROGRAMACION

A continuación se verán algunos de los comandos mas usados en la programación.

El comando **DO** repite un proceso el número de veces que se le indique.

073F7 DO *Empieza el bucle **DO** requiere un valor final y otro inicial.*
07334 LOOP *Termina el bucle **DO**.*

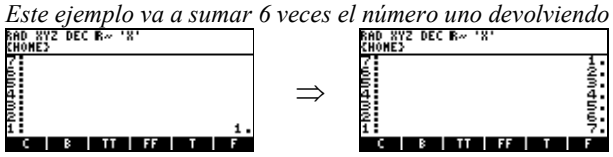
SINTAXIS

```
::
#final #inicial
DO
PROGRAMA
LOOP
;
@
```

Ejemplo:

```
::
 6 0
 DO
 CK1
 DUP
 %1
 %+
 LOOP
;
@
```

Este ejemplo va a sumar 6 veces el número uno devolviendo estos valores a la pila.



NOTA: *Este ejemplo requiere un número en el nivel 1, no puede haber ninguna otra cosa que no sea un número por ejemplo no puede haber "". Ya que esto ocasionaría la perdida de la memoria, para resolver este problema se puede hacer lo siguiente.*

```
::
CK1&Dispatch
real
 ::
  6 0
  DO
  DUP
  %1
  %+
  LOOP
 ;
;
@
```

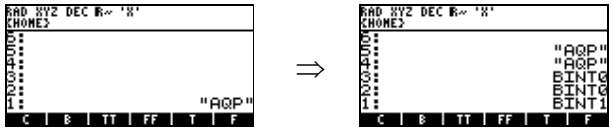
18.1 - PALABRAS PROPORCIONADAS AL BUCLE DO

Se proporcionan las siguientes palabras para usarlas con los bucles **DO**.

073A5 +LOOP
Es parecido a **LOOP** excepto que toma un entero binario de la pila e incrementa el contador del bucle en esa cantidad en lugar de en 1.

3645A DUPINDEX@
Duplica el objeto del nivel uno de la pila durante el número de veces que indique el número final
Ejemplo:

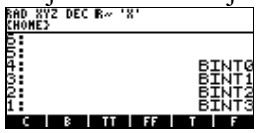
```
::
  2 0
  DO
    DUPINDEX@
  LOOP
;
@
```



07221 INDEX@
Devuelve el índice del entorno DoLoop mas interno:
Ejemplo:

```
::
  4 0
  DO
   INDEX@
  LOOP
;
@
```

Si ejecutamos este ejemplo devolverá los siguientes números enteros binarios.



367D9 INDEX@#-
Resta el número binario del nivel uno de la pila al índice del entorno DoLoop mas interno.

07270 INDEXSTO
Almacena un número binario del nivel uno de la pila como el índice del entorno DoLoop mas interno.

07249 ISTOP@
Devuelve el valor final del entorno DoLoop mas interno

07295 ISTOPSTO
Almacena un número binario del nivel uno de la pila como el valor de parada en el entorno DoLoop mas interno.

07258 JINDEX@
Devuelve el índice del segundo entorno DoLoop

07334 LOOP
Termina una estructura de bucle **DO**.

36482 OVERINDEX@
Hace **OVER** y luego devuelve el valor del índice del entorno DoLoop mas interno.

		➔	**4:**	Objeto 1
			3:	Objeto 2
2:	Objeto 1		**2:**	Objeto 1
1:	Objeto 2		**1:**	# indice

3646E SWAPINDEX@
Hace **SWAP** y luego devuelve el valor del índice del entorno DoLoop mas interno.

			3:	Objeto 2
2:	Objeto 1	➔	**2:**	Objeto 1
1:	Objeto 2		**1:**	# indice

3709B ZEROISTOPSTO
Almacena cero como el valor de parada del entorno DoLoop mas interno

364E1 toLEN_DO ({list} --> {list})
Comienza el bucle **DO** con inicio = #1 y con el valor de parada = #numero de elementos de la lista+1

34A22 IT
Requiere un flag si el flag es **TRUE** entonces ejecuta el programa **P1** y luego el programa **P2.**
Si es **FALSE** ejecuta **P2**.

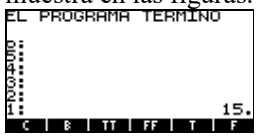
SINTAXIS:

```
::
flag
IT
 ::
  PROGRAMA P1
 ;
 ::
  PROGRAMA P2
 ;
;
@
```

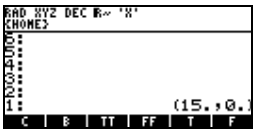
Ejemplo:

```
::
DUPTYPEREAL?
IT
 ::                          } P1
   %0
   %>C%
 ;
 ::
 "EL PROGRAMA TERMINO''      }
   FlashMsg                  } P2
 ;
;
@
```

Si introducimos el número real 15. y ejecutamos el programa, entonces mostrara el mensaje "**EL PROGRAMA TERMINO**" y convertirá el número real 15. en un número complejo como se muestra en las figuras.



⇒



34B3E ITE

Requiere un flag si el flag es **TRUE** entonces ejecuta el programa **P1** y termina, Si es **FALSE** ejecuta **P2**. y termina, esto vendría a ser parecido como un IF/THEN/ELSE del RPL de usuario.

SINTAXIS:

```
::
flag
ITE
 ::
  PROGRAMA P1
 ;
 ::
  PROGRAMA P2
 ;
;
@
```

Ejemplo:

```
::
DUPTYPEREAL?
ITE
  ::
   %0                    } P1
   %>C%                  }
  ;
  ::
   "NO ES UN NUMERO REAL"}
   FlashMsg              } P2
  ;
;
@
```

*Este ejemplo pregunta si es un número real, si es un número real devolverá el flag **TRUE** y ejecutara P1 colocando el número real 0 en la pila y después convirtiendo los dos números reales a complejos. Por ejemplo si en la pila teníamos el número real **45** devolverá **(45, 0)**.*
*Si el flag que devuelve es **FALSE** entonces ejecutara el programa P2 mostrando un mensaje.*

070C3 RPITE
Este comando tiene la siguiente sintaxis.

SINTAXIS:

```
::
flag
 ::
  programa 1
 ;
 ::
  programa 2
 ;
RPITE
;
@
```

Si el flag es **TRUE** se elimina flag y programa 2 y se evalúa programa 1.
Si el flag es **FALSE** se elimina flag y programa 1 y se evalúa programa 2.

Ejemplo:

```
::
DUPTYPEREAL?
  ::
    %0
    %>C%
  ;
  ::
   "EL PROGRAMA TERMINO''
    FlashMsg
  ;
RPITE
;
@
```

070FD RPIT
Este comando requiere un flag si el flag es **TRUE** entonces se elimina flag y se evalúa programa 1, si no, simplemente se eliminan flag y programa 1 y termina.

SINTAXIS:

```
::
flag
 ::  programa 1  ;
RPIT
;
@
```

Ejemplo:

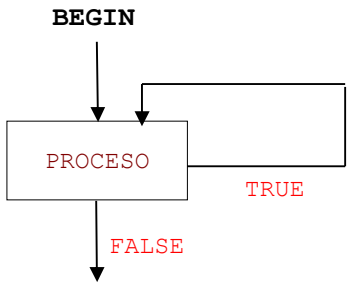
```
::
DUPTYPEREAL?
  ::
    %0
    %>C%
  ;
RPIT
;
@
```

18.2 BUCLES INDEFINIDOS

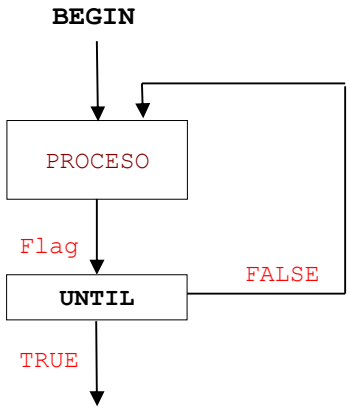
Los bucles indefinidos se construyen combinando las siguientes palabras RPL:

071A2 BEGIN
Copia el puntero del intérprete en la pila de retornos.

0716B IDUP
Igual que **BEGIN**.

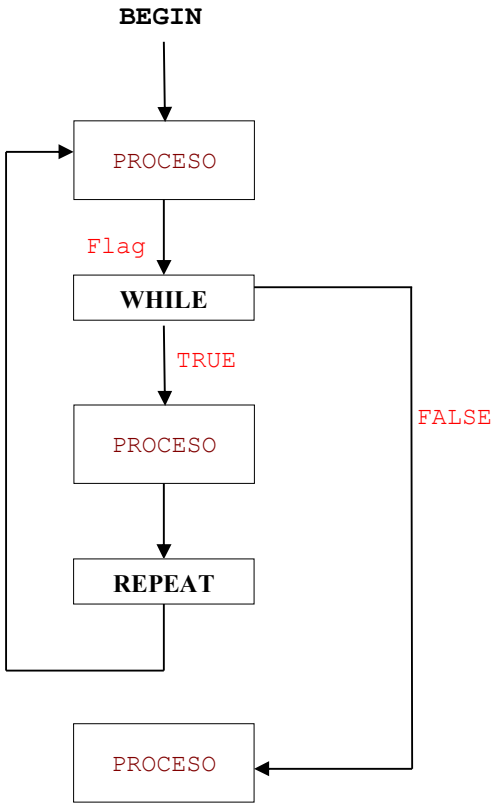


071C8 UNTIL
Requiere de un flag si flag es **TRUE** continua con la ejecución del programa.
Si el flag es **FALSE** vuelve a ejecutar lo que este a continuación de BEGIN.



071EE WHILE
Si flag es **TRUE**, entonces no hace nada. Si es **FALSE** elimina el primer puntero de la pila de retornos y hace que el puntero del interprete se salte los dos siguientes objetos.

071E5 REPEAT
Copia el primer puntero de la pila de retornos al puntero del intérprete



071AB AGAIN
El bucle BEGIN … AGAIN no tiene test para que se termine, en este bucle deberá ocurrir un error o que se manipule directamente la pila de retornos.

19. VARIABLES TEMPORALES

Las variables temporales se nombran dentro de una lista con el correspondiente identificador temporal en la lista usada por **BIND**.

Los nombres de la lista se usan en el mismo orden en que los objetos unidos aparecen en la pila el último identificador de la lista corresponde al objeto en el nivel 1, el identificador anterior al ultimo corresponde al objeto en el nivel 2 y así sucesivamente.

Ejemplo:
En este ejemplo el entero binario **ONE** se une a **Var1** y **TWO** se une a **Var2**

```
::
 ONE TWO
 {
  LAM Var1
  LAM Var2
 }
 BIND                    ( Une ONE con la variable temporal Var1 y TWO con la variable temporal Var2 )
 LAM Var1                ( Llama a ONE desde Var1 )
 LAM Var2                ( Llama a TWO desde Var2 )
  ' LAM Var1 STO         ( Almacena un nuevo objeto en Var1 )
 ABND                    ( Abandona el entorno temporal )
;
@
```

NOTA: ' *Esto indica que el siguiente comando no se evalúa solo se pone en la pila como si fuera un RCL.*

Los identificadores temporales pueden contener cualquier carácter de texto excepto que no deberías empezar los nombres con **'** o **#** ya que tales nombres están reservados para los programas incorporados en ROM.

Si no existe ninguna posibilidad que se cree otro entorno temporal encima del entorno que vas a crear, se pueden usar los nombres nulos para ahorrar memoria. Hay varias palabras de utilidades que te permiten acceder a variables locales en el entorno superior mediante su número de posición, que es más rápido que la resolución ordinaria de un nombre.

Por ejemplo, el ejemplo anterior seria así:

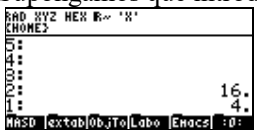
```
::
 ONE TWO
 { NULLLAM NULLLAM }
 BIND                    ( Une ONE y TWO a dos variables temporales con  nombres nulos )
 2GETLAM                 ( Llama a ONE desde la primera variable )
 1GETLAM                 ( Llama a TWO desde la ultima variable )
 2PUTLAM                 ( Almacena un nuevo objeto en la primera variable )
 ABND                    ( Abandona el entorno temporal )
;
@
```

La numeración comienza con la ultima variable temporal (o sea, en el mismo orden que el numero del nivel de la pila).

A continuación se verán 2 ejemplos para comprender el uso de variables temporales con nombre y variables temporales sin nombre.

En los ejemplos que se verán se deberán introducir dos números reales y al ejecutar el programa este devolverá la suma, resta y división.

Supongamos que introducimos los números reales 16 y 4.



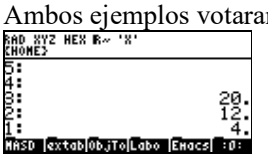
Ejemplo 1:

```
::
CK2&Dispatch
#11
 ::
  { LAM NUMERO1
    LAM NUMERO2
  }
  BIND
  LAM NUMERO1
  LAM NUMERO2
  %+
  LAM NUMERO1
  LAM NUMERO2
  %-
  LAM NUMERO1
  LAM NUMERO2
  %/
  ABND
 ;
;
@
```

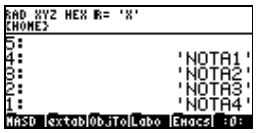
Ejemplo 2:

```
::
CK2&Dispatch
#11
 ::
 { NULLLAM NULLLAM  } BIND
 2 GETLAM
 1 GETLAM
 %+
 2 GETLAM
 1 GETLAM
 %-
 2 GETLAM
 1 GETLAM
 %/
 ABND
 ;
;
@
```

Ambos ejemplos votaran el mismo resultado, el que se muestra en la figura.



EJEMPLO DE COMO FUNCIONA GETLAM



4 GETLAM = NOTA1
3 GETLAM = NOTA2
2 GETLAM = NOTA3
1 GETLAM = NOTA4

Si en algún caso vamos a usar varias variables temporales como por ejemplo:
{ NULLLAM NULLLAM NULLLAM NULLLAM NULLLAM NULLLAM } BIND

Esto mismo se puede realizar de forma mas compacta de modo que ocupe menos espacio:
NULLLAM SIX NDUPN
SIX {}N BIND

Si queremos sobrescribir alguna variable temporal sin nombre se deberá usar el comando **PUTLAM**, por ejemplo si queremos cambiar el valor de **NOTA2** se deberá usar **3 PUTLAM.**

A continuación se muestran los comandos que operan con variables temporales:

35DEE 1ABNDSWAP (objeto ➔ lamob objeto)
Hace **`:: 1GETLAM ABND SWAP ;`**

364FF 1GETABND (➔ lamob)
Hace **`:: 1GETLAM ABND ;`**

34616 1GETLAM
(➔ ob)

346E8 22GETLAM
Devuelve el contenido del enesimo lam

35F42 1GETSWAP (objeto ➔ lamob objeto)
Hace **`:: 1GETLAM SWAP ;`**

36518 1LAMBIND (objeto ➔)
Hace **`:: 1NULLLAM{} BIND ;`**

2B3A6 1NULLLAM{} (➔ { NULLLAM })
Devuelve una lista con un lam nulo

34611 1PUTLAM (objeto ➔)
Almacena un objeto en la primera variable temporal sin nombre.

346E3 22PUTLAM
Almacena un objeto en el enesimo lam

3632E 2GETEVAL (➔ ?)
Llama y evalúa el objeto en el segundo lam

07943 @LAM (id ➔ objeto TRUE)
(id ➔ FALSE)
Llama **lam** por nombre, devuelve el objeto y **TRUE** si **id** existe; si no devuelve **FALSE**

07497 ABND (➔)
Abandona el entorno de variables temporales.

074D0 BIND (objeto ... { id ... } ➔)
Crea un nuevo entorno de variables temporales.

34D00 CACHE (objeton ... objeto1 n lam ➔)
Salva n objetos mas la cuenta n en un entorno temporal, estando unido cada objeto al mismo identificador lam. El último par tiene la cuenta.

34EBE DUMP (NULLLAM ➔ ob1..obn n)
DUMP es esencialmente el inverso de **CACHE**, Pero funciona cuando el nombre cacheado es **NULLLAM** y siempre hace una recolección de basura.

36513 DUP1LAMBIND (objeto ➔ objeto)
Hace **DUP** y luego **1LAMBIND**

34797 DUP4PUTLAM (objeto ➔ objeto)
Hace **DUP** y luego **4PUTLAM**

347AB DUPTEMPENV
Duplica el entorno temporal de la cima y borra la palabra de protección.

075A5 GETLAM (#n ➔ objeto)
Devuelve el objeto de la enésima variable temporal

2B3AB NULLLAM (➔ NULLLAM)
Nombre de variable temporal nulo

075E9 PUTLAM (objeto #n ➔)
Almacena un nuevo objeto en la enésima variable temporal

07D1B STOLAM (objeto id ➔)
Almacena un nuevo objeto en la variable temporal con nombre.

20 - VARIABLES Y DIRECTORIOS

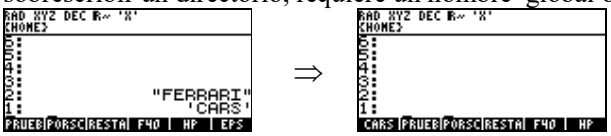
El fundamento en RPL del sistema de las palabras de usuario **STO** y **RCL** son las palabras **STO**, **CREATE** y **@**:

08696 CREATE

Crea en el directorio actual un directorio o programa dependiendo del contenido del objeto, se requiere el objeto en el nivel 2 de la pila y un ID (Nombre) en el nivel 1 de la pila.

07D27 STO

Almacena el contenido del objeto del nivel 2 con un nombre en el directorio actual, es capaz de sobrescribir un directorio, requiere un nombre global o local en nivel 1, y el objeto en el nivel 2.

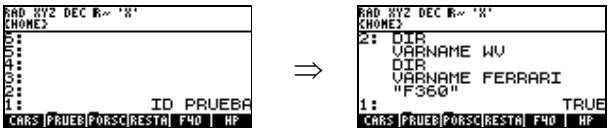


Nota: Esta palabra no es igual al comando de usuario STO.

0797B @

En el caso lam (Variables temporales), @ intenta emparejar lam a la parte objeto identificador temporal de una unión en el área de Entornos Temporales (buscando desde la primera área de entornos temporales (la mas interna) hacia la última). Si lo consigue, entonces el objeto unido al lam se devuelve junto con una bandera **TRUE**; si no, se devuelve una bandera **FALSE**.

En el caso id, @ intenta emparejar id con la parte nombre de una variable global, comenzando en el directorio actual y subiendo por los directorios padre si fuera necesario.
Si no se consigue resolver, entonces se devuelve una bandera **FALSE**.
Si se consigue resolver, se devuelve la parte objeto de la variable resuelta junto con una bandera **TRUE**.

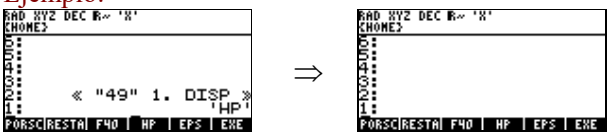


Una dificultad al usar **STO** y @ es que no hacen ninguna distinción con los comandos incorporados. Por esta razón, es mejor usar **SAFESTO** y **SAFE@**, que funcionan como **STO** y @ excepto que convierten automáticamente los cuerpos ROM en nombres XLIB (SAFESTO) y al revés de nuevo (SAFE@).

35A29 SAFESTO

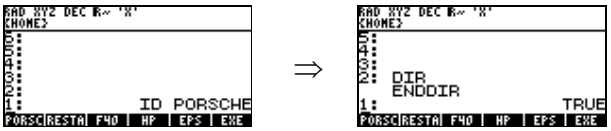
Almacena el objeto del nivel 2 de la pila con el nombre del nivel 1 de la pila. También es capaz de sobrescribir un directorio.

Ejemplo:



35A5B SAFE@

Requiere un nombre global o local, devuelve **TRUE** en el nivel 1 el contenido del nombre local o global en el nivel 2 de la pila si se encuentra el nombre, y devuelve **FALSE** en el nivel 1 si no se encuentra el nombre.



EXTENSIONES ADICIONALES

25E79 ?STO_HERE

Esto es la versión del sistema del **STO** de usuario. Es igual que **SAFESTO**, excepto que con las variables globales.

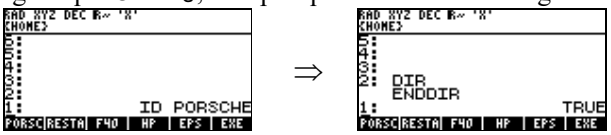
a) Almacena sólo en el directorio actual.

b) No sobrescribe un directorio.

25EF7 SAFE@_HERE

Requiere un nombre global o local, devuelve **TRUE** en el nivel 1 el contenido del nombre local o global en el nivel 2 de la pila si se encuentra el nombre, y devuelve **FALSE** en el nivel 1 si no se encuentra el nombre.

Igual que **SAFE@**, excepto que con los nombres globales la búsqueda se restringe al directorio actual.



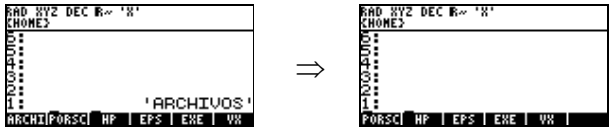
2F380 SysSTO

Guarda el objeto del nivel 2 en el directorio **HOME** requiere el nombre con el que será guardado en el nivel 1.

20.1 - OTRAS PALABRAS RELACIONADAS

08C27 PURGE

Elimina la variable especificada por un id, no hace ninguna comprobación de tipo en el objeto almacenado.

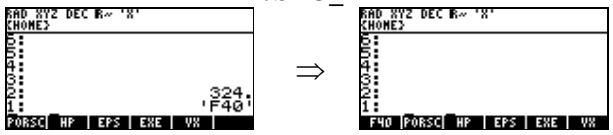


2F2A3 XEQRCL

Igual que `SAFE@` con las variables globales, pero produce un error si la variable no existe. (Trae el contenido de la variable que se indica en el nivel 1 de la pila.

25E79 XEQSTOID

Nombre alternativo de **?STO_HERE**

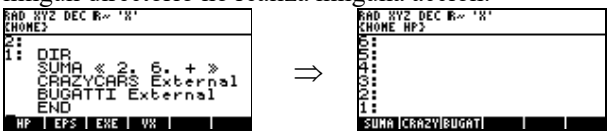


DIRECTORIOS.

Un directorio tiene la forma **DIR** *contenido del directorio* **END**. Un directorio vació simplemente tendrá la forma **DIR END.**

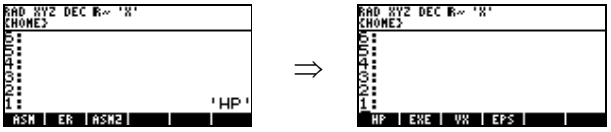
20.2 PALABRAS PARA LA MANIPULACIÓN DE DIRECTORIOS

08D08 CONTEXT!
Entra al directorio que tenga el contenido que se muestra en el nivel 1 de la pila. En este caso el contenido del directorio de nombre **HP** es el que se muestra y al ejecutar el programa entra al directorio **HP**. Si el contenido del directorio que se muestra en el nivel 1 de la pila no corresponde a ningún directorio no realiza ninguna acción.

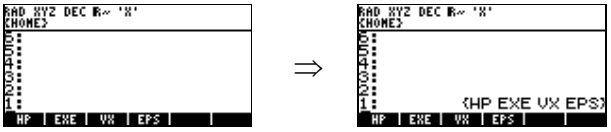


08D5A CONTEXT@
Muestra en el nivel 1 de la pila el contenido del directorio actual

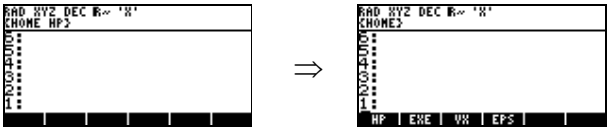
25EA1 CREATEDIR
Crea un directorio en el directorio actual.



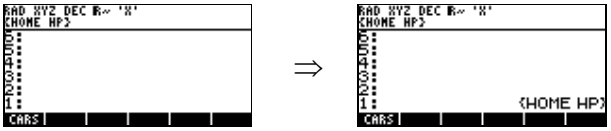
25EB9 DOVARS
Devuelve una lista con los nombres de las variables del directorio actual.



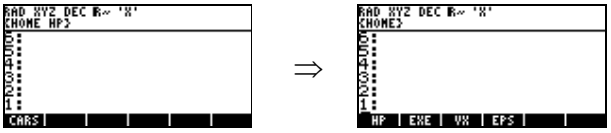
08D92 HOMEDIR
Hace de HOME el directorio actual.



25EF1 PATHDIR
Devuelve el camino actual.

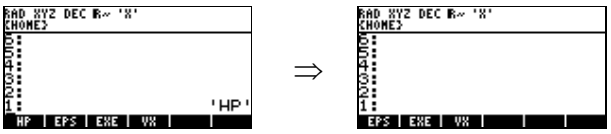


2F265 UPDIR
Sube un directorio.



2F296 XEQORDER
Ordena el directorio actual, requiere una lista con el orden de los objetos.

25F14 XEQPGDIR
Elimina un directorio mientras respeta las convenciones de referencia/recolección de basura. Requiere el nombre del directorio a eliminar.



077E4 CRDIR# (#libnum ➔ rrp)
Devuelve un directorio vacío en la pila, requiere un numero #.

08DF2 !CREATEDIR (id ➔)
Crea un directorio vacío, requiere como entrada un id, no verifica si el nombre esta en uso.
Es lo equivalente a:

```
:: # 7FF CRDIR# SWAP CREATE ;
```

082E3 RAM-WORDNAME (ob ➔ id)
Devuelve el id de un objeto.

2F296 XEQORDER ({id1 id2..} ➔)
Ordena las variables del directorio según como este en una lista.

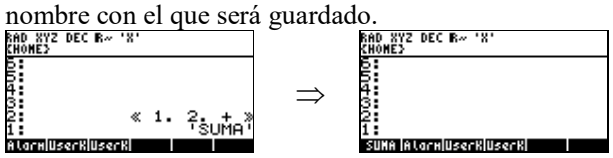
20.3 EL DIRECTORIO OCULTO

La HP49G al igual que la HP48 tiene un directorio oculto en el cual se guardan algunas variables **Alarms**, **UserKeys** y **UserKeys.crc** algunos programas usan este directorio para almacenar algunas variables.

Las palabras que veremos a continuación proporcionan capacidades de almacenamiento, llamada y eliminación en el directorio oculto:

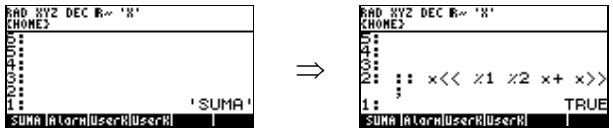
37104 StoHiddenVar

Almacena un objeto en el directorio oculto, para esto se debe colocar el contenido del objeto y el nombre con el que será guardado.



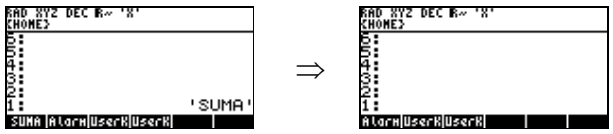
370AF RclHiddenVar

Llama al contenido de una variable que este almacenada en el directorio oculto, para esto se debe poner el nombre de la variable, si la variable existe devolverá el contenido de la variable y la bandera **TRUE**, de lo contrario devolverá la bandera **FALSE**.



37118 PuHiddenVar

Elimina una variable oculta para esto se debe poner el nombre de la variable a eliminar en el nivel 1.

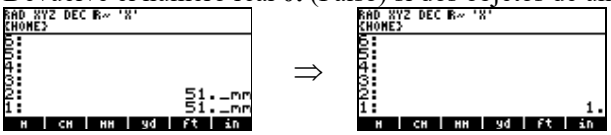


21 OBJETOS DE UNIDADES

A continuación se verán algunos comandos para manejar unidades.

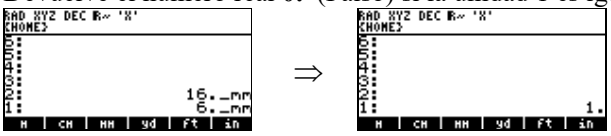
2F087 UM=?

Devuelve el número real **1.** (Verdadero) si dos objetos de unidades son iguales.
Devuelve el número real **0.** (Falso) si dos objetos de unidades son diferentes.



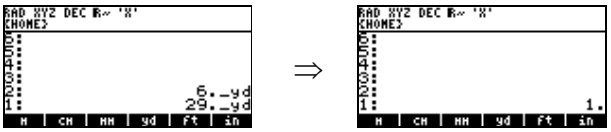
2F07C UM#?

Devuelve el número real **1.** (Verdadero) si la unidad 1 es diferente a la unidad 2.
Devuelve el número real **0.** (Falso) si la unidad 1 es igual a la unidad 2.



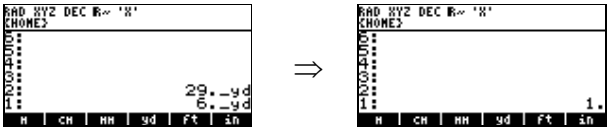
2F086 UM<?

Devuelve el número real **1.** (Verdadero) si la unidad 1 es menor a la unidad 2.
Devuelve el número real **0.** (Falso) si la unidad 1 es mayor a la unidad 2.



2F089 UM>?

Devuelve el número real **1.** (Verdadero) si la unidad 1 es mayor a la unidad 2.
Devuelve el número real **0.** (Falso) si la unidad 1 es menor a la unidad 2.



2F085 UM<=?

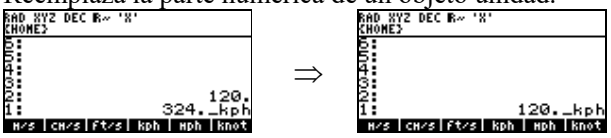
Devuelve el número real **1.** (Verdadero) si la unidad 1 es menor o igual a la unidad 2.
Devuelve el número real **0.** (Falso) si la unidad 1 es mayor a la unidad 2.

2F088 UM>=?

Devuelve el número real **1.** (Verdadero) si la unidad 1 es mayor o igual a la unidad 2.
Devuelve el número real **0.** (Falso) si la unidad 1 es menor a la unidad 2.

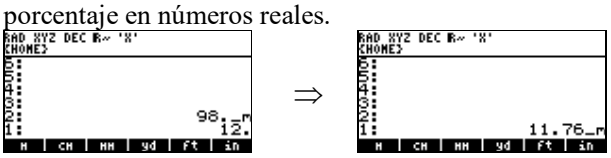
2F07A UM>U

Reemplaza la parte numérica de un objeto unidad.



2F07D UM%

Devuelve el porcentaje del objeto unidad que esta en el nivel 2 de la pila. En el nivel 1 se indica el porcentaje en números reales.



En este ejemplo saca el 12% de 98 metros.

2F07E UM%CH

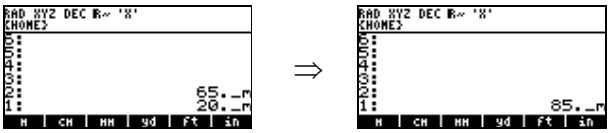
Devuelve la diferencia porcentual de la unidad 2 respecto a la unidad 1 en números reales.

2F07F UM%T

Devuelve la fracción porcentual en números reales.

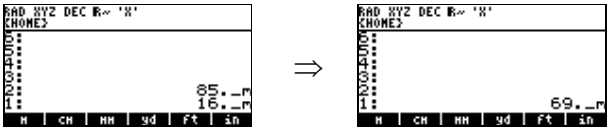
2F081 UM+

Suma la unidad 1 con la unidad 2.



2F082 UM-

Resta la unidad 1 con la unidad 2.

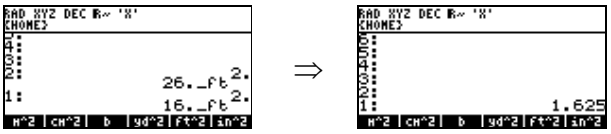


2F080 UM*

Multiplica la unidad 1 con la unidad 2.

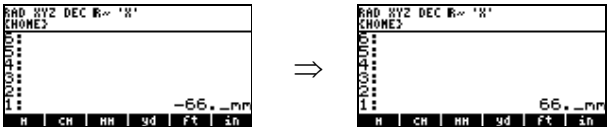
2F083 UM/

Divide la unidad 1 con la unidad 2.



2F08A UMABS

Valor absoluto de la unidad del nivel 1.

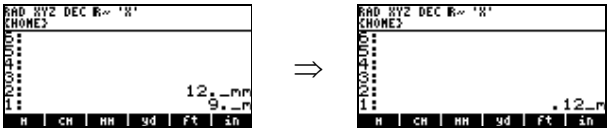


2F08B UMCHS

Cambio de signo al objeto unidad.

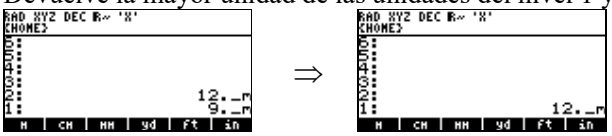
2F08C UMCONV

Convierte las unidades del nivel 2 a las unidades del nivel 1.



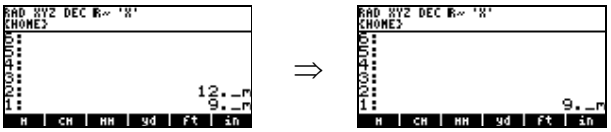
2F08E UMMAX

Devuelve la mayor unidad de las unidades del nivel 1 y2.



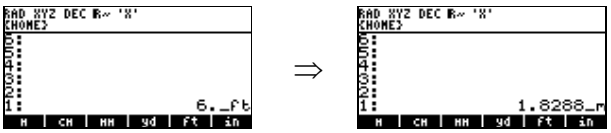
2F08F UMMIM

Devuelve la menor unidad de las unidades de los niveles 1 y 2.



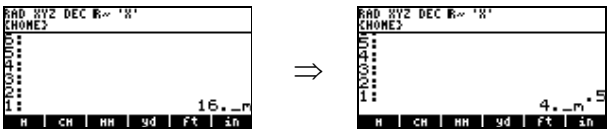
2F090 UMSI

Convierte a las unidades del nivel uno a las unidades del Sistema Internacional (SI)



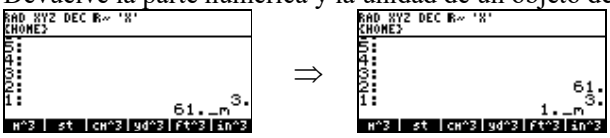
2F093 UMSQRT

Saca la raíz cuadrada a la unidad del nivel 1.



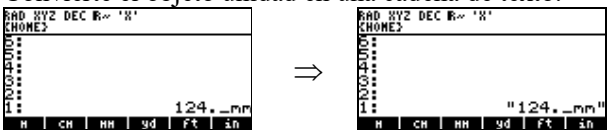
2F095 UMU>

Devuelve la parte numérica y la unidad de un objeto de unidad que se encuentre en el nivel 1.



2F019 UNIT>$

Convierte el objeto unidad en una cadena de texto.

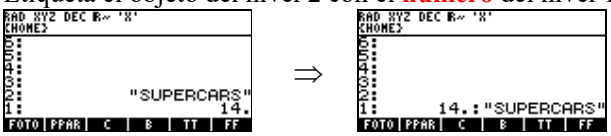


22 - OBJETOS ETIQUETADOS

Están disponibles las siguientes palabras para manipular objetos etiquetados.
Recuerda que un objeto puede tener etiquetas múltiples.

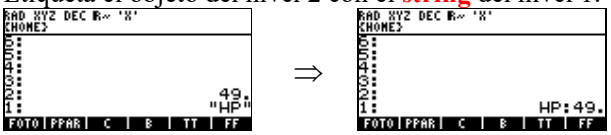
2F223 %>TAG

Etiqueta el objeto del nivel 2 con el **número** del nivel 1.



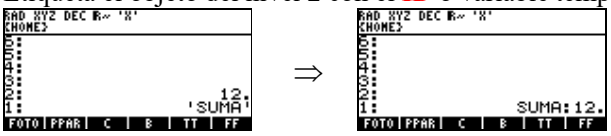
05E81 >TAG

Etiqueta el objeto del nivel 2 con el **string** del nivel 1.



05F2E ID>TAG

Etiqueta el objeto del nivel 2 con el **ID** o variable temporal del nivel 1.

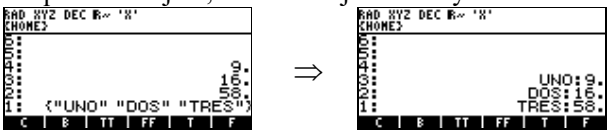


37ABE STRIPTAGS

Quita las etiquetas del objeto del nivel 1.

37B04 TAGOBS

Etiqueta un objeto, o varios objetos si hay una lista de etiquetas en el nivel 1



22 - HERRAMIENTAS DE ERROR

Las siguientes palabras se suministran para el control de errores:

04E5E ERRSET
Inicia la captura de errores.

04EB8 ERRTRAP
Si ocurre un error ejecuta otro programa.

SINTAXIS:

```
::
 ERRSET
  :: PROGRAMA A SER EXAMINADO ;
 ERRTRAP
  :: PROGRAMA A EJECUTAR EN CASO DE QUE OCURRA UN ERROR ;
;
```

Ejemplo:

```
::
 ERRSET
  ::
   CK1
   %1
   %+
  ;
 ERRTRAP
  ::
   "COLOCA CUALQUIER NUMERO"
   FlashMsg
   %1
   dowait
 ;
;
@
```

26067 ERRBEEP
Genera un pitido de error.

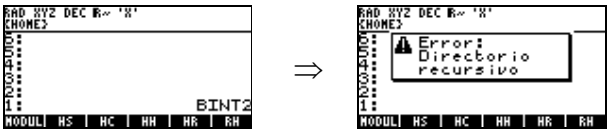
2EE61 FlashWarning
Genera un pitido de error y muestra un mensaje de error.

Ejemplo:

```
::
 "ACABA DE OCURRIR UN ERROR"
 FlashWarning
;
@
```

04EA4 ABORT
Hace **ERRORCLR** y **ERRJMP** es parecido al comando de usuario **KILL**.

25EAE DO#EXIT
Almacena un nuevo número de error y ejecuta **ERRJMP**; también ejecuta **AtUserStack,** requiere como entrada el número del error en enteros binarios.



3619E 'ERRJMP
Pone el objeto **ERRJMP** en la pila

04ED1 ERRJMP
Invoca al subsistema de control de errores.

04CE6 ERROR@
Devuelve el número de error actual en enteros binarios.

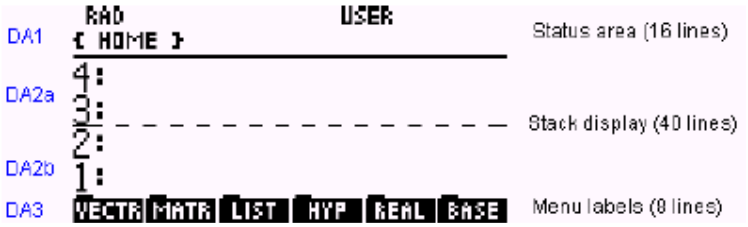
04D33 ERRORCLR
Almacena cero como el número de error

36883 ERROROUT
Almacena un nuevo numero de error y hace **ERRJMP,** requiere como entrada el número de error en enteros binarios.

04D0E ERRORSTO
Almacena un nuevo numero de error, requiere como entrada el número de error en enteros binarios.

04EB8 ERRTRAP
Se salta el siguiente objeto del "runstream".

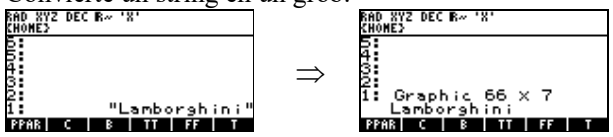
23 - LA PANTALLA



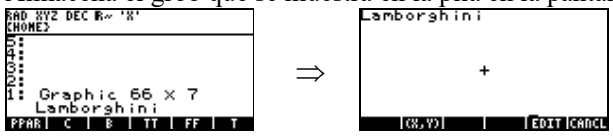
2EE69	**SetDA1Temp**	*Congela el Area de la pantalla 1*
2EE8A	**SetDA2aTemp**	*Congela el Area de la pantalla 2a*
2EE6A	**SetDA2bTemp**	*Congela el Area de la pantalla 2b*
2F37A	**SetDA2OKTemp**	*Congela el Area de la pantalla 2a y 2b.*
2EE6B	**SetDA3Temp**	*Congela el Area de la pantalla 3.*
2EE71	**SetDA12Temp**	*Congela el Area de la pantalla 1 y 2*
2EE64	**SetDAsTemp**	*Congela toda el Area de la pantalla*
2EE8D	**ClrDA1OK**	*Redibuja el Area de la pantalla 1*
2EE8E	**ClrDA2aOK**	*Redibuja el Area de la pantalla 2a*
2EE8F	**ClrDA2bOK**	*Redibuja el Area de la pantalla 2b*
2EE90	**ClrDA2OK**	*Redibuja el Area de la pantalla 2a y 2b*
2EE6E	**ClrDA3OK**	*Redibuja el Area de la pantalla 3*
2EE6D	**ClrDAsOK**	*Redibuja toda el Area de la pantalla.*
2EF5E	**BlankDA1**	*Borra el Area de la pantalla 1*
2F31B	**BlankDA2**	*Borra el Area de la pantalla 2*
2EE5C	**BlankDA12**	*Borra el Area de la pantalla 1y 2*
261C5	**CLEARLCD**	*Borra toda el Area de la pantalla.*

23.1 GRAFICOS Y PANTALLA

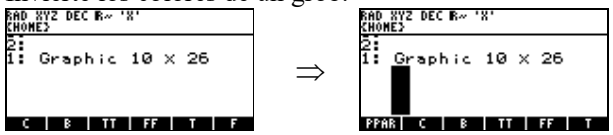
25F7C $>GROB
Convierte un string en un grob.



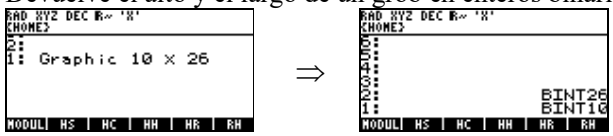
25ED8 GROB>GDISP
Almacena el grob que se muestra en la pila en la pantalla grafica. (grob gráfico)



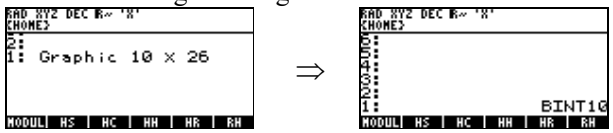
2609E INVGROB
Invierte los colores de un grob.



26085 GROBDIM
Devuelve el alto y el largo de un grob en enteros binarios.



36C68 GROBDIMw
Devuelve el largo de un grob en enteros binarios.



2607B GROB!
Almacena el grob1 dentro del grob2. (grob1 grob2 #x #y ➔)

2F342 GROB+#
Inserta el grob 2 dentro de la posición especificada del grob1, usando **OR** si flag es **TRUE** o **XOR** si flag es **FALSE**. (flag grob1 grob2 #x #y ➔ grob')

2F0DB MAKEPICT# (#w #h ➔)
Crea un grob grafico en blanco. El tamaño mínimo debe ser 131x64.si el grob es mas pequeño entonces automáticamente lo redimensiona.

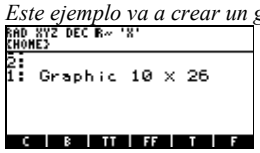
2F324 CKGROBFITS
Disminuye el grob2 si no cabe (encaja) en el grob1.
(grob1 grob2 #n #m ➔ grob1 grob2' #n #m)

2F320 CHECKHEIGHT (grob #alto ➔)
Fuerza al grob (ABUFF/GBUFF) a tener menos de 64 filas de alto.

260B2 MAKEGROB
Crea un grob en blanco, para esto se necesita dar el alto y largo del grob en enteros binarios.
Ejemplo:

```
::
26
10
MAKEGROB
;
@
```

*Este ejemplo va a crear un grob en blanco de tamaño **10** de largo **x 26** de alto.*



25F0E XYGROBDISP
Muestra un grob en pantalla requiere de 3 argumentos, las coordenadas de #x y #y y el grob a mostrar.

26080 GROB!ZERO
Pone a cero una sección rectangular del grob. Requiere el grob y las coordenadas.
(grob #x1 #y1 #x2 #y2 ➔ grob')

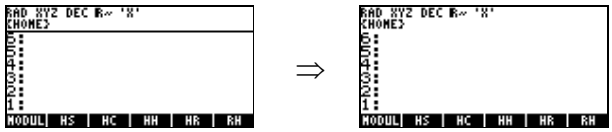
368E7 GROB!ZERODRP
Pone a cero una sección rectangular de un grob. Requiere el grob y las coordenadas que se van a poner a cero. (grob #x1 #y1 #x2 #y2)

2608F HARDBUFF
Devuelve el grob de la pantalla actual (➔ HBgrob (el grob de pantalla actual))

26099 HEIGHTENGROB
Añade #filas al grob, a menos que el grob sea nulo, requiere el grob y el número de filas a agregar.
(grob #filas)

2EFA0 LINEOFF
Borra los píxeles en una línea del grob de texto. Requiere las coordenadas (#x1 #y1 #x2 #y2)
Nota: #x2 debe ser mayor que #x1 (usar ORDERXY#)
Ejemplo:

```
::
 BINT0
 BINT14
 BINT131
 BINT14
 LINEOFF
 SetDA1Temp
;
@
```

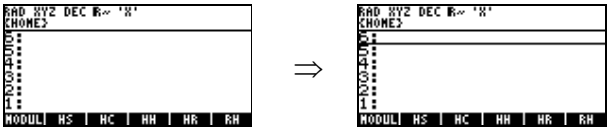


2EFA3 LINEOFF3
Borra los píxeles en una línea del grob gráfico. Requiere las coordenadas (#x1 #y1 #x2 #y2)
Nota: #x2 debe ser mayor que #x1 (usar ORDERXY#)
Ejemplo:

```
::
 BINT0
 BINT14
 BINT131
 BINT14
 LINEOFF3
;
@
```

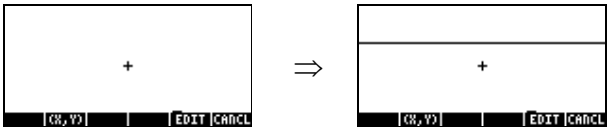
2EF9F LINEON
Activa los píxeles en una línea del grob de texto. Requiere las coordenadas (#x1 #y1 #x2 #y2)
Nota: #x2 debe ser mayor que #x1 (usar ORDERXY#)
Ejemplo:

```
::
 BINT0
 BINT20
 BINT131
 BINT20
 LINEON
 SetDAsTemp
;
@
```



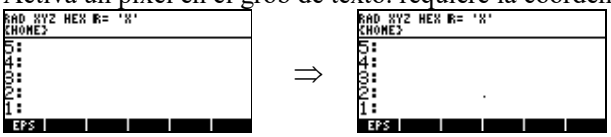
2EFA2 LINEON3
Activa los píxeles en una línea del grob gráfico. Requiere las coordenadas (#x1 #y1 #x2 #y2)
Nota: #x2 debe ser mayor que #x1 (usar ORDERXY#)
Ejemplo:

```
::
 BINT0
 BINT20
 BINT131
 BINT20
 LINEON3
;
@
```



280C1 ORDERXY#
Ordena dos puntos para dibujar una línea, requiere las coordenadas (#x1 #y1 #x2 #y2), los ordena y devuelve las nuevas coordenadas ya ordenada (#x1 #y1 #x2 #y2)

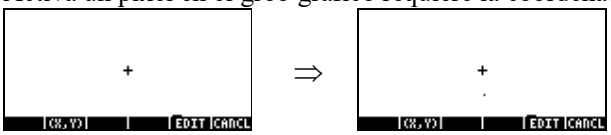
260E4 PIXON
Activa un píxel en el grob de texto. requiere la coordenada (#x #y) del píxel a activar.



260DF PIXOFF
Borra un píxel en el grob de texto, requiere la coordenada (#x #y) del píxel a borrar.

260EE PIXON?
Devuelve **TRUE** si el píxel del grob de texto esta activado. requiere la coordenada del píxel a preguntar.
(#x #y ➔ flag)

260DA PIXON3
Activa un píxel en el grob gráfico requiere la coordenada del píxel (punto) (#x #y)



260D5 PIXOFF3
Borra un píxel en el grob gráfico requiere la coordenada (#x #y) del píxel a borrar.

260E9 PIXON?3
Devuelve **TRUE** si el píxel del grob gráfico esta activado, requiere la coordenada del píxel a preguntar.
(#x #y ➔ flag)

255BF PixonB
Dibuja un píxel de color negro.
Requiere un grob y la coordenada (#x1 #y1).

255BA PixonW
Dibuja un píxel de color blanco.
Requiere un grob y la coordenada (#x1 #y1).

255C4 PixonG1
Dibuja un píxel de color gris ligero.
Requiere un grob y la coordenada (#x1 #y1).

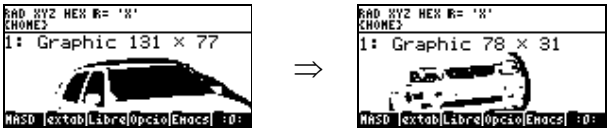
255C9 PixonG2
Dibuja un píxel de color gris oscuro.
Requiere un grob y la coordenada (#x1 #y1).

255CE PixonXor
Dibuja un píxel invirtiendo el color del píxel.
Requiere un grob y la coordenada (#x1 #y1).

2612F SUBGROB (grob #x1 #y1 #x2 #y2 ➔ subgrob)
Corta una parte de un grob requiere el grob y las coordenadas (#x1 #y1 #x2 #y2) que van a ser recortar para crear el nuevo grob.
Por ejemplo asumiendo que tenemos el siguiente grob en el nivel 1 de la pila.

Si ejecutamos el programa:

Entonces obtenemos:



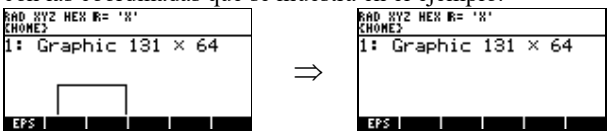
2EFA1 TOGLINE
Dibuja una línea inviertiendo los colores (pixeles) del grob de texto.
Requiere las coordenadas de la línea. (#x1 #y1 #x2 #y2)
#x2 debe ser mayor que #x1 para dibujar la línea

2EFA4 TOGLINE3
Dibuja una línea invirtiendo los colores (pixeles) del grob gráfico.
Requiere las coordenadas de la línea. (#x1 #y1 #x2 #y2)
#x2 debe ser mayor que #x1 para dibujar la línea

255EC LBoxG1
Dibuja un rectángulo de color gris ligero , requiere un grob y las coodernadas (#x1 #y1 #x2 #y2)
Ejemplo:

```
::
12 16
49 45
LBoxG1
;
@
```

Nota: Se supone que antes de ejecutar el programa se esta colocando un grob que tiene dibujado un rectangulo con las coordinadas que se muestra en el ejemplo.



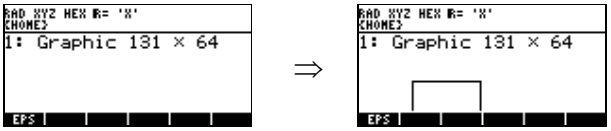
27AA3 NULLGROB_
Dibuja un grob nulo de 0 x 0.

280C1 ORDERXY#
Ordena los números enteros binarios (#x1 #y1 #x2 #y2), para dibujar un grob.

280F8 ORDERXY%
Ordena los números reales (%x1 %y1 %x2 %y2), para dibujar un grob.

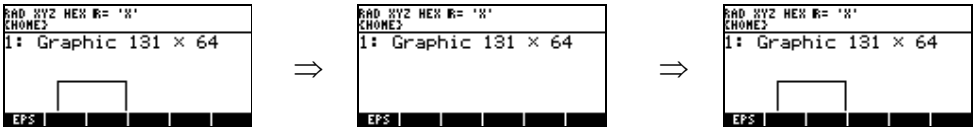
255F1 LBoxG2
Dibuja un rectángulo de color gris oscuro, requiere un grob y las coordenadas (#x1 #y1 #x2 #y2).
Ejemplo:

```
::
12 16
49 45
LBoxG2
;
@
```



255FB LBoxXor
Dibuja un rectángulo invirtiendo los colores (Pixeles).
Requiere un grob y las coordenadas (#x1 #y1 #x2 #y2).
Ejemplo:

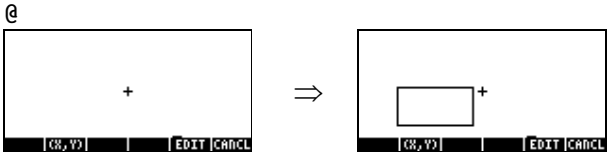
```
::
 12 16
 49 45
 LBoxXor
;
@
```



255A1 Grey?
Devuelve **TRUE** si el grob es de escala de grises. De lo contrario **FALSE**. Requiere un grob.

2F32C DRAWBOX#
Dibuja un rectangulo en el grob grafico requiere las coordenadas (#x1 #y1 #x2 #y2)
Ejemplo:

```
::
 20 30
 60 50
 DRAWBOX#
;
@
```



2F218 CircleW
Dibuja un circulo de color blanco. Requiere un grob la coordenada de #x de #y y el radio #r del circulo. Todo en números enteros binarios.
Ejemplo:

```
::
 64 131 MAKEGROB INVGROB
 50 25
 15
 CircleW
;
@
```

2F216 CircleG1
Dibuja un circulo de color gris ligero. Requiere un grob la coordenada de #x de #y y el radio #r del circulo. Todo en números enteros binarios.

2F217 CircleG2
Dibuja un circulo de color gris oscuro. Requiere un grob la coordenada de #x de #y y el radio #r del circulo. Todo en números enteros binarios.

2F215 CircleB
Dibuja un circulo de color negro. Requiere un grob la coordenada de #x de #y y el radio #r del circulo. Todo en números enteros binarios.
Ejemplo:

```
::
 64 131 MAKEGROB
 50 25
 14
 CircleB
;
@
```

2F219 CircleXor
Dibuja un circulo invirtiendo los colores de los pixeles. Requiere un grob la coordenada de #x de #y y el radio #r del circulo. Todo en números enteros binarios.

255E2 FBoxXor
Dibuja un rectangulo relleno invirtiendo los colores de los pixeles.
Requiere un grob y las coordenadas (#x1 #y1 #x2 #y2).

255F6 LBoxB
Dibuja un rectangulo de color negro requiere un grob y las coordenadas (#x1 #y1 #x2 #y2).

255E7 LBoxW
Dibuja un rectangulo de color blanco requiere un grob y las coordenadas (#x1 #y1 #x2 #y2).

255D3 FBoxW
Dibuja un rectangulo rellenado de color blanco.
Requiere un grob y las coordenadas (#x1 #y1 #x2 #y2).
Ejemplo:

```
::
 64 131 MAKEGROB INVGROB
 20 10
 60 30
 PTR 255D3
;
@
```

255DD FBoxB
Dibuja un rectangulo rellenado de color negro.
Requiere un grob y las coordenadas (#x1 #y1 #x2 #y2).
Ejemplo:

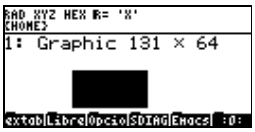
```
::
 64 131 MAKEGROB
 20 10
 60 30
 FBoxB
;
@
```

255D3 FBoxG1
Dibuja un rectangulo rellenado de color gris ligero.
Requiere un grob y las coordenadas (#x1 #y1 #x2 #y2).
Ejemplo:

```
::
 64 131 MAKEGROB INVGROB
 20 10
 60 30
 FBoxG1
;
@
```

255D8 FBoxG2
Dibuja un rectangulo rellenado de color gris oscuro.
Requiere un grob y las coordenadas (#x1 #y1 #x2 #y2).
Ejemplo:

```
::
 64 131 MAKEGROB
 20 10
 60 30
 FBoxG2
;
@
```



23.2 COMO DESPLAZAR UN GROB

Comandos que sirven para mover la pantalla mientras se mantienen pulsadas las teclas.	
2F36D SCROLLDOWN	Desplaza la pantalla un píxel hacia abajo.
2F36E SCROLLLEFT	Desplaza la pantalla un píxel hacia la izquierda.
2F36F SCROLLRIGHT	Desplaza la pantalla un píxel hacia la derecha.
2F370 SCROLLUP	Desplaza la pantalla un píxel hacia arriba.

Comandos que sirven para mover la pantalla directamente hacia un extremo.	
2F347 JUMPBOT	Desplaza la pantalla directamente hacia abajo.
2F348 JUMPLEFT	Desplaza la pantalla directamente hacia la izquierda.
2F349 JUMPRIGHT	Desplaza la pantalla directamente hacia la derecha.
2F34A JUMPTOP	Desplaza la pantalla directamente hacia arriba.

Comandos que sirven para mover la pantalla	
26193 WINDOWUP	Desplaza la pantalla un píxel hacia arriba
26184 WINDOWDOWN	Desplaza la pantalla un píxel hacia abajo.
26189 WINDOWLEFT	Desplaza la pantalla un píxel hacia la izquierda.
2618E WINDOWRIGHT	Desplaza la pantalla un píxel hacia la derecha.

Comandos que sirven para preguntar sobre el estado de la pantalla	
2F38D WINDOWTOP?	Devuelve un flag indicando si la pantalla esta desplazada totalmente hacia arriba.
2F38A WINDOWBOT?	Devuelve un flag indicando si la pantalla esta desplazada totalmente hacia abajo.
2F38B WINDOWLEFT?	Devuelve un flag indicando si la pantalla esta desplazada totalmente hacia la izquierda
2F38C WINDOWRIGHT?	Devuelve un flag indicando si la pantalla esta desplazada totalmente hacia la derecha.

24 - MENUS

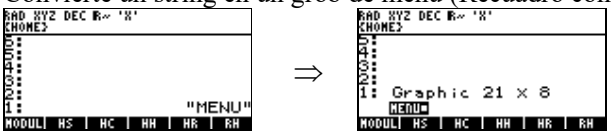
A continuación se verán algunos comandos que sirven para hacer menús.
Al referirnos a un grob de menú nos referimos a un grob de dimensiones 21 x 8.

En el siguiente cuadro se verán las coordenadas de los menús en x.

MENÚS	COORDENADA DEL MENÚ (ENTEROS BINARIOS)
Menú 1 (F1)	0
Menú 2 (F2)	22
Menú 3 (F3)	44
Menú 4 (F4)	66
Menú 5 (F5)	88
Menú 6 (F6)	110

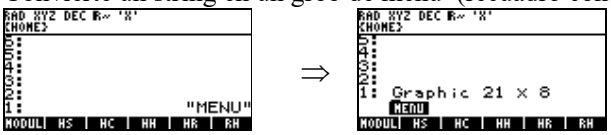
2E189 MakeBoxLabel

Convierte un string en un grob de menú (Recuadro con una cuadradito en su interior).



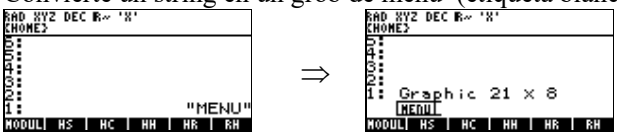
2E1EB MakeDirLabel

Convierte un string en un grob de menú (recuadro con una barra de directorio)



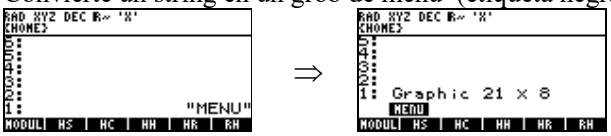
2E24D MakeInvLabel

Convierte un string en un grob de menú (etiqueta blanca)



2E166 MakeStdLabel

Convierte un string en un grob de menú (etiqueta negra estándar)



2E0F3 Str>Menu

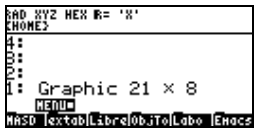
Convierte un string en un menú requiere la coordinada en x y el string.

Ejemplo:

```
::
0 "MENU"
Str>Menu
;
@
```

25E7F Box/StdLabel
Convierte un string en un grob de menú, requiere un string y un flag si el flag es **TRUE** dibuja un recuadro, si es **FALSE** no dibuja nada.
Ejemplo:

```
::
"MENU"
TRUE
Box/StdLabel
;
@
```

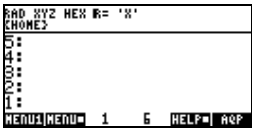


2E0D5 Grob>Menu
Muestra un grob de menú, requiere la coordenada en x y el grob de menú.
Ejemplo:

```
::
 22
 GROB 0003A 8000051000000000002000003000002000002000002000007000000000
 Grob>Menu
;
@
```

A continuación se vera un ejemplo donde se aplican todos los comandos anteriores.

```
::
  BINT0   "MENU1"  Str>Menu
  BINT22  "MENU2"  TRUE  Box/StdLabel  Grob>Menu
  BINT44
  GROB 0003A 8000051000000000002000003000002000002000002000007000000000
  Grob>Menu
  BINT66
  GROB 0003A 8000051000000000007000001000001000007000005000007000000000
  Grob>Menu
  BINT88  "HELP"
  TRUE  Box/StdLabel  Grob>Menu
  BINT110  "AQP"  Str>Menu
  WaitForKey
;
@
```



A continuación algunas palabras para el control de los menús:

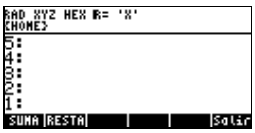
2DFF4 DispMenu.1
Actualiza y vuelve a mostrar el menú actual.

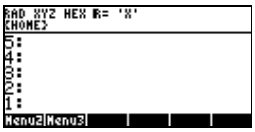
2DFE0 DispMenu
Actualiza y vuelve a mostrar el menú actual y también llama a **SetDA3Valid** para congelar la línea de menú de la pantalla.

27FED NullMenuKey
Inserta un menú en blanco.

25EE0 InitMenu
Inicializa un menú personalizado, este comando tiene la siguiente sintaxis:

SINTAXIS:
```
::
{ { "MENU 1" :: Programa 1 ; }
  { "MENU 2" :: Programa 2 ; }
  { "MENU 1" :: Programa n ; }
} InitMenu
;
@
```

Si se desea introducir un menú en blanco se puede hacer con el comando **NullMenuKey.**
Ejemplo:
```
::
{ { "SUMA" x+ } { "RESTA" x- } NullMenuKey NullMenuKey NullMenuKey
{ "Salir"  :: %0 InitMenu% ; } } InitMenu
;
@
```


25F00 StartMenu
Se usa para personalizar menús, requiere el menú y el número en el que empezara el menú.
Ejemplo:
```
::
{
 { "Menu1" :: "Ejecutaste menu1" FlashMsg  ; }
 { "Menu2" :: "Ejecutaste menu2" FlashMsg  ; }
 { "Menu3" :: "Ejecutaste menu3" FlashMsg  ; }
}
2
StartMenu
;
@
```


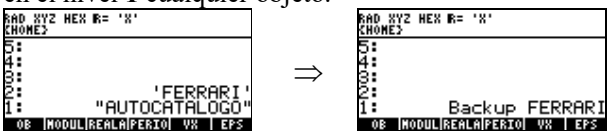
***Nota**: en este ejemplo el 2 indica que empiece a mostrar a partir del menú 2. al presionar NXT, aparecerán todos los menús.*

En ambos casos se puede incluir un grob de menú en lugar de las etiquetas ($).

25 - BACKUPS

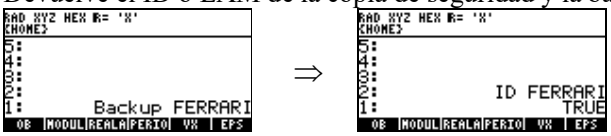
2F255 OB>BAK

Requiere de 2 argumentos en el nivel **2** un **ID** o **LAM** (El nombre que tendrá la copia de Seguridad) y en el nivel **1** cualquier objeto.



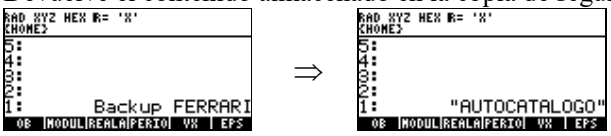
081D9 BAKNAME

Devuelve el ID o LAM de la copia de seguridad y la bandera **TRUE**.



0905F BAK>OB

Devuelve el contenido almacenado en la copia de seguridad.



26 - DoImputForm

Con este comando podemos crear menús como **MODE** o cuando entramos a resolver ecuaciones. Este comando es un poco complejo requiere de lo siguiente:

Etiquetas
Campos
 # de Etiquetas
 # de Campos
MessageHandler (*Normalmente* **'DROPFALSE**)
Titulo
DoImputForm

Etiquetas

Cada etiqueta consta de 3 Argumentos:

Nombre de la Etiqueta	(*En un String " "*)
Coordenada en X	(*En números enteros binarios*)
Coordenada en Y	(*En números enteros binarios*)

Campos

Cada campo requiere de 13 Argumentos:

MessageHandler	(*Normalmente* **'DROPFALSE**)
Coordenada en X	(*En números enteros binarios*)
Coordenada en Y	(*Normalmente la coordenada de la* ***Etiqueta*** *en Y + 2*)
Longitud	(*La longitud que va a tener el campo en números enteros binarios*)
Altura	(*La altura que va a tener el campo normalmente es* **9**)
Tipo de Campo	(*El tipo de campo que se va a querer esto se describe mas abajo*)
{ Tipos de Objetos }	(*Los tipos de objeto que va a aceptar el campo*)
Descompilación	
" Ayuda "	(*La ayuda en un String " "*)
Datos Choose	
Descompilación Choose	
Valor de reinicio	(*Valor que aparecerá si se presiona RESET*)
Valor inicial	(*Valor inicial que aparecerá en el campo*)

Tipo de Campo

1	Campo de texto	**(DF/EDF)**
3	Campo Algebraico	**(DF)**
12	Lista de Campos	**(LF)**
32	Check	**(CF)**
44	Choose	

Tipo de Objeto

MINUSONE	Cualquier Objeto
0	Reales
1	Complejos
2	String
3	Formación real
4	Formación compleja
5	Lista
6	ID
8	Programa
9	Simbólicos
10	Entero Binario
8	Programa
13	Objeto Unidad

Descompilación
Modo Estándar ⇒ **4**
Choose o Lista de campos ⇒ **17**
MINUSONE ⇒ Cualquiera

Datos Choose
Si no hay datos en un CHOOSE o si no es un CHOOSE entonces **MINUSONE.**

Descompilación Choose
Si es un campo Choose se deberá colocar el entero binario **17** .
Si no es un campo Choose se deberá colocar **MINUSONE.**

Valor de reinicio
Si es DF/EDF ⇒ **MINUSONE**
Si es LF ⇒ **{"etiqueta" cuerpo}**
Si es CF ⇒ **TRUE/FALSE**

Valor inicial
Igual que en los valores de reinicio si no se desea ningún valor inicial entonces **MINUSONE**.

MessageHandler
Es un secundario que se pone en la pila pero que no se evalúa generalmente es **'DROPFALSE**

Titulo
El titulo en un string.

DoInputForm

Resultados del comando DoImputForm
La salida al stack o pila del usuario al pulsar **ENTER** es:
N+1: Campo 1
N: Campo 2
...
2: Campo N
1: **TRUE**

Si se pulsa **CANCL** retornara el flag **FALSE**.

Ejemplo 1:

```
::
  "MARCA"  BINT10  BINT12
  'DROPFALSE
  BINT1  BINT9
  BINT6  BINT9
  BINT32
  MINUSONE  MINUSONE
  "AYUDA"
  MINUSONE  MINUSONE
  FALSE  FALSE
  BINT1  BINT1
  'DROPFALSE
  "TITULO"
  DoInputForm
;
@
```

Ejemplo 2:

```
::
  "ELIGE"  BINT0  BINT15
  'DROPFALSE
  BINT24  BINT11
  BINT44  BINT9
  BINT12
  MINUSONE
  BINT17
  "AYUDA"
  {
    {  "PARTE 1"   :: "PROGRAMA 1"  ;  }
    {  "PARTE 2"   :: "PROGRAMA 2"  ;  }
    {  "PARTE N"   :: "PROGRAMA N"  ;  }
  }
  BINT17
    {  "PARTE 2"   :: "PROGRAMA 2"  ;  }
    {  "PARTE 2"   :: "PROGRAMA 2"  ;  }
  BINT1  BINT1
  'DROPFALSE
  "TITULO"
  DoInputForm
;
@
```

Ejemplo 3:

```
::
  * Definición de las etiquetas
  "EQ:"   BINT0   BINT11
  "VAR:"  BINT0   BINT22
  "TOL"   BINT70  BINT22
  "RAICES COMPLEJAS"  BINT10  BINT31
  "MODO DEL ANGULO"   BINT0  BINT41

  * Definicion del campo EQ
  'DROPFALSE
  BINT16  BINT7
  BINT107  BINT10
  BINT3
  MINUSONE
  BINT4
  "ENTRE UNA EQUACION"
  MINUSONE  MINUSONE
  MINUSONE  MINUSONE

  * Definición del campo VAR
  'DROPFALSE
  BINT16  BINT18
  BINT30  BINT9
  BINT1
  { BINT6 }
  BINT4
  ""
  MINUSONE  MINUSONE
  MINUSONE  MINUSONE

  * Definición del campo TOL
  'DROPFALSE
  BINT83  BINT18
  BINT40  BINT9
  BINT1
  { BINT0  }
  BINT4
  "INGRESA TOLERANCIA"
  MINUSONE  MINUSONE
  % .001
  % .001

  * Definicion del campo Checkmark
  'DROPFALSE
  BINT1  BINT28
  BINT6  BINT9
  BINT32
  MINUSONE  MINUSONE
  "RAICES COMPLEJAS?"
  MINUSONE  MINUSONE
  FALSE  FALSE
  * Definicion del campo ANGLE MODE
  'DROPFALSE
  BINT64  BINT37
  BINT30  BINT9
  BINT12
  MINUSONE
  BINT17
  "SELECCIONA MODO DEL ANGULO"
```

```
  {
    { "RAD"  xRAD  }
    { "DEG"  xDEG  }
  }
  BINT17
  { "RAD"  xRAD  }
  { "RAD"  xRAD  }
  * Información  General
  BINT5  BINT5
  'DROPFALSE
  "RESOLVER EQUACION"
  DoInputForm
;
@
```

27 - COMANDOS QUE OPERAN EN UN PROGRAMA

34AA1 ?SEMI
Requiere un flag si este es **TRUE** sale del programa en curso.

3692D ?SEMIDROP
Requiere un objeto en el nivel 2 y un flag en el nivel 1 de la pila, si el flag es **TRUE** elimina el objeto si el flag es **FALSE** sale del 1programa en curso.

0712A ?SKIP
Requiere un flag si este es **TRUE**, se salta el siguiente programa a continuación de **?SKIP**.

34A92 NOT?SEMI
Requiere un flag si este es **FALSE** sale del programa en curso.

0714D SKIP
Salta un objeto en el runstream.

0715C 2SKIP
Salta los dos siguientes objetos en el runstream.
Ejm:

*Al ejecutar el programa se salta **BINT1 BINT2** y solo devuelve **BINT3**.*

349F9 case
Case toma una bandera de la pila; si es **TRUE**, case ejecuta el objeto que le sigue en el runstream mientras baja la pila de retornos al puntero del interprete, descartando el resto del programa que sigue al objeto (como COLA). Si es **FALSE**, case se salta el siguiente objeto y continua con el programa (como **SKIP**).
SINTAXIS:

```
::
 flag case :: programa 1 ;
 flag case :: programa 2 ;
       .
       .
       .
 flag case :: programa n ;
 :: programa por defecto ;
;
@
```

NOTA: Si ninguno de los flags es **TRUE** entonces ejecuta programa por defecto, basta que un flag sea **TRUE** y se salta los demas objetos por ejemplo si el flag1 y el flag2 ambos son **TRUE** solo ejecuta el flag1 como se explica en la definición de **case.**

Ejemplo:

```
::
 40 TestSysFlag
 case
  :: 40 ClrSysFlag ;
  :: 40 SetSysFlag ;
;
@
```

Lo que hacen las palabras que se listan más abajo esta generalmente suficientemente claro a partir de sus nombres.

Los nombres tienen (hasta) tres partes: una parte inicial, luego "**case**" y luego una parte final. La parte inicial indica que se hace antes de la acción "**case**", o sea, "xxxcase..." es equivalente a "xxx case...".

Las palabras que tienen una parte final después de "**case**" son de dos tipos.

En un tipo, la parte final indica el objeto mismo ejecutado condicionalmente, o sea, "...caseyyy" es equivalente a "...case yyy".

En el otro tipo, la parte final es una palabra o palabras que se incorporan al siguiente objeto. caseDROP y casedrop son del primer y segundo tipo respectivamente. **caseDROP** es equivalente a **case DROP**; **casedrop** es como **case** con un **DROP** incorporado en el siguiente objeto.

Palabras que hacen **COLA** o **SKIP** al siguiente objeto:

34939 #=casedrop (# # →)
(# #' → #)
Se debería llamar OVER#=casedrop
Ejemplo:
En este ejemplo se deberá colocar un número entero binario en la pila.

```
::
 1 #=casedrop :: "numero 1" FlashMsg ;
 2 #=casedrop :: "numero 2" FlashMsg ;
 3 #=casedrop :: "numero 3" FlashMsg ;
 4 #=casedrop :: "numero 4" FlashMsg ;
 :: "numero mayor que 4" FlashMsg ;
;
@
```

2B1A3 %1=case (% →)

2B149 %0=case (% → flag)

36E6B ANDNOTcase (flag1 flag2 →)

36D4E ANDcase (flag1 flag2 →)

34985 case2drop (ob1 ob2 TRUE →)
(FALSE →)

03495 casedrop (ob TRUE →)
(FALSE →)

348F7 DUP#0=case (# → #)

3490E DUP#0=csedrp (# → #) # <> #0
(# →) # = #0

36E7F EQUALNOTcase (ob ob' →)

36D62 EQUALcase (ob ob' →)

36D08 EQUALcasedrp (ob ob' ob' →)
(ob ob' ob'' → ob)

34999 EQcase (ob1 ob2 →)

34A13 **NOTcase** (flag →)

3494E **NOTcasedrop** (ob FALSE →)
(TRUE →)

359E3 **ORcase** (flag1 flag2 →)

348E2 **OVER#=case** (# #' → #)

Palabras que o salen o continúan con el siguiente objeto:

36C4F **caseDoBadKey** (flag →) Sale vía DoBadKey

349D6 **case2DROP** (ob1 ob2 TRUE →)
(FALSE →)

349B1 **caseDROP** (ob TRUE →)
(FALSE →)

365E5 **caseFALSE** (TRUE → FALSE)
(FALSE →)

3652C **caseTRUE** (TRUE → TRUE)
(FALSE →)

365B3 **casedrpfls** (ob TRUE → FALSE)
(FALSE →)

365CC **case2drpfls** (ob1 ob2 TRUE → FALSE)
(FALSE →)

368FB **casedrptru** (ob TRUE → TRUE)
(FALSE →)

36D21 **DUP#0=csDROP** (#0 →)
(# → #) # <> 0.

36914 **NOTcaseTRUE** (FALSE → TRUE)
(TRUE →)

29E99 **tok=casedrop** ($ $' → :: ob1 ;)
($ $' → :: $ ob2 rest ;)

34ABE **ITE_DROP** (ob T → :: ob2 rest ;)
(ob F → :: ob ob1 rest ;)

36F65 **UserITE** (#set → :: ob1 ob3 rest ;)
(#clr → :: ob2 rest ;)

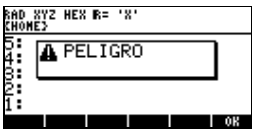
36F79 **SysITE** (#set → :: ob1 ob3 rest ;)
(#clr → :: ob2 rest ;)

28 - MENSAJES TEMPORALES

A veces es conveniente mostrar una advertencia y luego devolver la pantalla a su estado previo. Hay varias técnicas y herramientas disponibles para esto. El modo mas fácil de hacerlo es con la palabra FlashWarning.

2EE61 FlashWarning

Muestra un mensaje de error y emite un pitido de error.
Ejemplo:
:: "PELIGRO" FlashWarning **;**



NOTA: *Es importante usar* ***DISPROW1**** *y* ***DISPROW2**** *en vez de* ***DISPROW1*** *y* ***DISPROW2*** *si existe la posibilidad que HARDBUFF haya sido desplazado.*
No existen las palabras correspondientes para otras líneas de la pantalla.

29 - UBICACION DE LAS TECLAS

La palabra de usuario **WAIT** devuelve un número real que es esta codificado de la forma rc.p donde:

r = La fila de la tecla
c = La columna de la tecla
p = El plano de desplazamiento

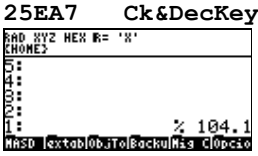
Entre los planos de desplazamiento tenemos los planos 1, 2, 3, 4, 5 y 6 que se describen en las siguientes 2 tablas.

P	PLANOS PRINCIPALES
1	Sin desplazar
2	
3	

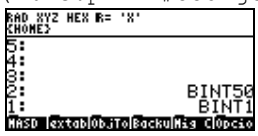
p	PLANOS ALFA
4	
5	
6	

La traducción entre la numeración interna de las teclas y la numeración rc.p se puede llevar a cabo con dos palabras:

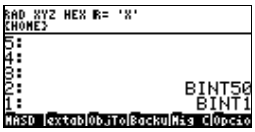
25EA7 Ck&DecKeyLoc (%rc.p ➔ #CódigoTecla #Plano)



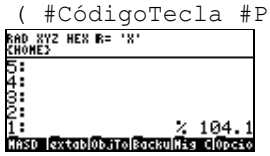
⇒



25EA9 CodePl>%rc.p (#CódigoTecla #Plano ➔ %rc.p)



⇒



29.1 - ESPERAR UNA TECLA

Si una aplicación necesita esperar a que se pulse una sola tecla, lo mejor es usar la palabra **WaitForKey**, que devuelve la tecla pulsada en el formato completo (tecla + plano).

WaitForKey también controla los indicadores alfa y desplazamiento así como el procesado de alarmas.

Están disponibles las siguientes palabras:

04708 CHECKKEY (➔ #CódigoTecla TRUE)
(➔ FALSE)

Devuelve, pero no la saca, la siguiente tecla en el bufer.

261CA FLUSHKEYS
Vacía el búfer de teclado

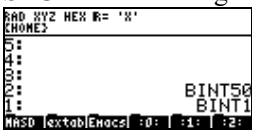
04714 GETTOUCH (➔ #CódigoTecla TRUE)
(➔ FALSE)
Devuelve y saca la siguiente tecla del búfer.

25EE3 KEYINBUFFER?
Devuelve **TRUE** si hay una tecla en el búfer de teclado, si no devuelve **FALSE**.

25FAE ATTN?
Devuelve **TRUE** si se ha pulsado [ATTN] de lo contrario devuelve **FALSE**.

05068 ATTNFLGCLR
Borra la bandera de la tecla attn (no saca la tecla attn del búfer)

25F0B WaitForKey (➔ #CódigoTecla #Plano)
Devuelve la siguiente tecla que se pulse en formato completo (tecla + plano) Por ejemplo si pulsamos **SPC** devuelve el siguiente código:



30 - EL BUCLE EXTERNO PARAMETRIZADO

En esta sección, el termino "bucle externo parametrizado" se usa para referirse al uso de la palabra RPL "ParOuterLoop" o al uso combinado de las utilidades fundamentales que lo componen, todo lo cual se puede ver como palabras que toman el control del teclado y de la pantalla hasta que se cumple una condición especificada.

El bucle externo parametrizado, "ParOuterLoop", toma nueve argumentos en este orden:

AppDisplay
El objeto de actualización de la pantalla que se evalúa antes de la evaluación de cada tecla. "AppDisplay" debería controlar la actualización de la pantalla no controlada por las teclas mismas y también debería realizar un control especial de errores.

AppKeys
Las asignaciones de las teclas físicas, un objeto secundario con el formato que se describe más abajo.

NonAppKeyOK?
Una bandera *(Flag)* que especifica si las teclas físicas no asignadas por la aplicación deben realizar sus acciones por defecto o ser canceladas.

DoStdKeys?
Una bandera que se usa conjuntamente con "**NonAppKeyOK?**" que especifica si las teclas que no usa la aplicación usan las definiciones estándar de las teclas en lugar del procesado de teclas por defecto.

AppMenu
La especificación de las teclas de menú, un secundario o una lista con el formato especificado en el documento de asignaciones de teclas de menú, o **FALSE**

#AppMenuRow
El número de fila del menú inicial de la aplicación. En la mayoría de las aplicaciones, debe ser el entero binario uno.

SuspendOK?
Una bandera que especifica si cualquier comando de usuario que crearía un entorno suspendido y restauraría el bucle externo del sistema debería generar en su lugar un error o no.

ExitCond
Un objeto que se evalúa a **TRUE** cuando se va a salir del bucle externo o si no, **FALSE**. "ExitCond" se evalúa antes de cada actualización de la pantalla y de cada evaluación de tecla de la aplicación.

AppError
El objeto controlador de errores que se evalúa si se produce un error durante la parte del bucle externo parametrizado de evaluación de una tecla.

El bucle externo parametrizado mismo no devuelve ningún resultado.
Sin embargo, cualquiera de las teclas usadas por la aplicación puede devolver resultados a la pila de datos o de cualquier manera que se quiera.

30.1 - UTILIDADES DEL BUCLE EXTERNO PARAMETRIZADO

La palabra del bucle externo parametrizado "ParOuterLoop" consiste enteramente en llamadas (con el adecuado control de errores) a sus cuatro palabras de utilidades RPL, que son, en orden:

2B4AC POLSaveUI
Salva la actual interfase de usuario (UI) en un entorno temporal. No toma argumentos y no devuelve ningún resultado.

2B542 POLSetUI
Dispone (configura) el interfase de usuario actual de acuerdo a los mismos par metros que requiere el "ParOuterLoop". No devuelve ningún resultado.

2B628 POLKeyUI
Muestra, lee y evalúa teclas, controla errores y sale de acuerdo a la interfase de usuario especificado por "POLStetUI". No toma ningún argumento ni retorna ningún resultado.

2B6CD POLRestoreUI
Restaura la interfase de usuario salvado por **"POLSaveUI"** y abandona el entorno temporal.
No toma ningún argumento ni devuelve ningún resultado.

Además de las cuatro utilidades de arriba, se usa la utilidad:

2B6B4 POLResUI&Err
Para proteger el interfase de usuario salvado en el caso de un error que no se controla dentro del bucle externo parametrizado.

Estas utilidades pueden usar las aplicaciones que requieren un mayor control sobre la interfase de usuario. Por ejemplo:

- Para una ejecución óptima, una aplicación puede crear un entorno temporal con variables temporales con nombres nulos después de llamar a **"POLSaveUI"**, luego acceder a las variables con nombres nulos "dentro" de **"POLKeyUI"**, ya que solo **"POLSaveUI"** crea un entorno temporal del bucle externo parametrizado y solo **"POLRestoreUI"** accede al mismo entorno.

- Para evitar gastos innecesarios y que consumen tiempo, una aplicación que usa múltiples bucles externos parametrizados consecutivos (no anidados) puede llamar a **"POLSaveUI"** al principio de la aplicación, luego llamar a **"POLSetUI"** y a **"POLKeyUI"** múltiples veces a lo largo de la aplicación y finalmente llamar a **"POLRestoreUI"** al final de la aplicación.

30.2 EXAMEN DEL BUCLE EXTERNO PARAMETRIZADO

El bucle externo parametrizado opera como se esboza a continuación.

2B4AC POLSaveUI
Salva la interfase del sistema o del usuario de la aplicación actual

Si se produce un error

2B542 POLSetUI
Configura la interfase de usuario de la nueva aplicación

2B628 POLKeyUI
Mientras "**ExitCond**" se evalúe a **FALSE**

{ Evalúa "**AppDisplay**" Si se produce un error Lee y evalúa una tecla
Entonces Evalúa "AppError" }

Entonces Restaura el interfase de usuario y **ERRJMP**

2B6CD POLRestoreUI
Restaura la interfase de usuario salvado

El bucle externo parametrizado crea un entorno temporal cuando salva la interfase de usuario actual y abandona este entorno cuando restaura una interfase de usuario salvado. Esto significa **que las palabras que operan en el entorno temporal mas reciente, tales como "1GETLAM", NO se deberían usar "dentro" del bucle externo parametrizado** (o sea, en una definición de tecla o en el objeto de actualización de la pantalla de la aplicación) **A MENOS QUE el entorno temporal deseado se cree DESPUES DE llamar a "POLSaveUI"** y se abandone antes de llamar a "**POLRestoreUI**".

En los entornos temporales creados antes de llamar al bucle externo parametrizado, las aplicaciones deberían crear y operar con variables temporales con nombre.

30.3 CONTROLAR ERRORES CON LAS UTILIDADES

Para asegurar que puede restaurar adecuadamente una interfase de usuario si se produce un error dentro de una aplicación, el bucle externo parametrizado protege la interfase de usuario salvado estableciendo una trampa de error inmediatamente después de su llamada a "**POLSaveUI**", como se muestra a continuación:

```
::
   POLSaveUI              ( salva el interfase de usuario actual )
   ERRSET                 ( prepara la restauración del interfase de
                            usuario salvado en caso de error )
   ::
     POLSetUI             ( Configura el interfase de usuario de la
                            Aplicación )
     POLKeyUI             ( muestra, lee y evalúa )
   ;
   ERRTRAP                ( si error, entonces restaura el interfase de
                            usuario salvado y da error )
   POLResUI&Err
   POLRestoreUI           ( restaura el interfase de usuario salvado )
;
@
```

El propósito de la utilidad soportada "**POLResUI&Err**" es restaurar el interfase de usuario salvado por "**POLSaveUI**" y luego dar error.

Las aplicaciones que usan las utilidades del bucle externo parametrizado en vez del "**ParOuterLoop**" tienen que incluir este mismo nivel de protección del interfase de usuario salvado en el control de errores.

30.4 ASIGNACIONES DE TECLAS FISICAS

Cualquier tecla en cualquiera de los 6 planos puede ser asignada durante la ejecución del bucle externo parametrizado.
El parámetro del bucle "**AppKeys**" especifica las teclas a asignar y sus nuevas asignaciones.

Si se quiere un control total "**NonAppKeyOK?**" debe ser **FALSE.**

Para no dar tanta teoria vamos al asunto.

SINTAXIS:
Aca se pueden asignar cualquiera de los 6 planos.

```
:: Plano 1  #=casedrop
   :: Codigo Tecla 1 ?CaseKeyDef    :: Programa a ejecutar ;
      Codigo Tecla 2 ?CaseKeyDef    :: Programa a ejecutar ;
            .
            .
            .
      Codigo Tecla n ?CaseKeyDef    :: Programa a ejecutar ;
   ;
   Plano 2  #=casedrop
   :: Codigo Tecla 1 ?CaseKeyDef    :: Programa a ejecutar ;
      Codigo Tecla 2 ?CaseKeyDef    :: Programa a ejecutar ;
            .
            .
            .
      Codigo Tecla n ?CaseKeyDef    :: Programa a ejecutar ;
   ;
            .
            .
            .
   Plano n  #=casedrop
   :: Codigo Tecla 1 ?CaseKeyDef    :: Programa a ejecutar ;
      Codigo Tecla 2 ?CaseKeyDef    :: Programa a ejecutar ;
            .
            .
            .
      Codigo Tecla n ?CaseKeyDef    :: Programa a ejecutar ;
   ;
;
```

30.5 ASIGNACIONES DE TECLAS DE MENÚS

Solo se pueden asignar las teclas de menós (F1 F2 F3 F4 F5 F6) en cualquiera de los tres planos (Normal,  y ) para que se inicialice cuando se inicia el bucle externo parametrizado.

El parámetro del bucle externo "**AppMenu**" especifica el objeto de inicialización **(una lista o un secundario)** para el menú de la aplicación o **FALSE**, indicando que se dejar intacto el menú actual. Cuando se sale del bucle externo parametrizado, se restaura automáticamente el menú anterior.

Si "AppMenu" **es una lista nula**, entonces se hace un conjunto de seis asignaciones de teclas nulas de menú.

Si "AppMenu" **es FALSE**, entonces se mantiene el menú presente cuando se llama al bucle externo parametrizado.

***NOTA*:** *las asignaciones de **teclas fisicas** (teclas "duras"/hardkeys) tienen prioridad sobre las asignaciones de **teclas de menú** (teclas "blandas"/softkeys). Esto significa que el controlador de teclas físicas debe incluir la siguiente línea si se van a procesar teclas de menú:*

```
DUP#<7 casedrpfls
```

El parametro **AppMenu** tiene la siguiente **SINTAXIS:**

```
{
 { "Etiqueta Menú 1" {
                     :: TakeOver ( Programa Plano Normal ) ;
                     :: TakeOver ( Programa Plano  ) ;
                     :: TakeOver ( Programa Plano  ) ;
                     }
 }
 { "Etiqueta Menú 2" {
                     :: TakeOver ( Programa Plano Normal ) ;
                     :: TakeOver ( Programa Plano  ) ;
                     :: TakeOver ( Programa Plano  ) ;
                     }
 }
          .
          .
          .
{ "Etiqueta Menú 6" {
                     :: TakeOver ( Programa Plano Normal ) ;
                     :: TakeOver ( Programa Plano  ) ;
                     :: TakeOver ( Programa Plano  ) ;
                     }
 }
}
```

La Etiqueta puede ser cualquier objeto, pero normalmente es una cadena de caracteres (String) o un grob de 8x21. (grob de menú).

La palabra **NullMenuKey** inserta una tecla de menú *(Etiqueta)* en blanco que pita cuando se pulsa.

30.6 EVITAR ENTORNOS SUSPENDIDOS

Una aplicación puede tener que permitir evaluar comandos y objetos de usuario arbitrarios, pero no querer que el entorno actual sea suspendido por los comandos "**HALT**" o "**PROMPT**".

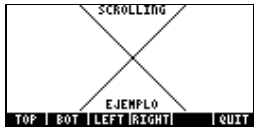
Si el parámetro del bucle externo "**SuspendOKa?**" es **FALSE**, entonces cualquier comando que suspendería el entorno genera el error "**HALT not Allowed**" (HALT no Permitido), permitiendo que "**AppError**" lo maneje.

Si "**SuspendOK?**" es **TRUE**, entonces la aplicación debe estar preparada para controlar las consecuencias. Aqui los peligros son muchos y graves.

En todas la aplicaciones previsibles, "**SupendOK?**" debe ser **FALSE.**

Ejemplo 1: ParOuterLoop

```
::
   RECLAIMDISP                      ( Llama la pantalla alfa )
   ClrDA1IsStat                     ( Desactiva temporalmente el reloj )
   ZEROZERO                         ( #0 #0 )
   150 150 MAKEGROB                 ( #0 #0 150x150grob )
   XYGROBDISP                       ( Crea un grob )
   ZEROZERO                         ( #x1 #y1 )
   149 149                          ( #x1 #y1 #x2 #y2 )
   LINEON                           ( Dibuja una línea )
   ZERO 149                         ( #x1 #y1 )
   149 ZERO                         ( #x1 #y1 #x2 #y2 )
   LINEON                           ( Dibuja una línea )
   HARDBUFF
   75 50 "SCROLLING"                ( HBgrob 75 150 "SCROLLING" )
   150 CENTER$3x5                   ( HBgrob )
   75 100 "EJEMPLO"                 ( HBgrob 75 100 "EXAMPLE" )
   150 CENTER$3x5                   ( HBgrob )
   DROPFALSE                        ( FALSE )
   { LAM Exit } BIND                ( Une la bandera de salida del POL )
   '
    DispMenu.1                      ( Muestra y actualiza el menú)
  '                                 ( Controlador de las teclas físicas )
    ::
       1  #=casedrop                ( Plano Normal )
        ::
          DUP#<7 casedrpfls         ( habilita las teclas de menús )
          10     ?CaseKeyDef   :: TakeOver SCROLLUP ;
          14     ?CaseKeyDef   :: TakeOver SCROLLLEFT ;
          15     ?CaseKeyDef   :: TakeOver SCROLLDOWN ;
          16     ?CaseKeyDef   :: TakeOver SCROLLRIGHT ;
          47     ?CaseKeyDef   :: TakeOver TRUE ' LAM Exit STO ;
        ;
     2DROP 'DoBadKeyT
    ;
   TrueTrue                          ( Banderas del control de teclado )
   {
    { "TOP"   :: TakeOver JUMPTOP ; }
    { "BOT"   :: TakeOver JUMPBOT ; }
    { "LEFT"  :: TakeOver JUMPLEFT ; }
    { "RIGHT" :: TakeOver JUMPRIGHT ; }
    NullMenuKey
    { "QUIT" :: TakeOver TRUE ' LAM Exit STO ; }
    }
ONEFALSE                            ( Primera fila, no suspender )
' LAM Exit                          ( Condición de salida de la aplicación)
' ERRJMP                           ( Controlador de Errores )
ParOuterLoop                        ( Ejecuta el ParOuterLoop )
RECLAIMDISP                         ( Cambia el Tamaño y borra la pantalla)
;
@
```



Ejemplo 2: ParOuterLoop

En este ejemplo tenemos guardado un grob de tamaño 167 x 89 en un archivo con el nombre FOTO.

```
::
  RECLAIMDISP
  ZEROZERO
  ID FOTO
  XYGROBDISP
  FALSE
  { LAM EXIT }
  BIND
  ' NOP
  '
  ::
    1 #=casedrop
    ::
      10 ?CaseKeyDef   :: TakeOver SCROLLUP  ;
      14 ?CaseKeyDef   :: TakeOver SCROLLLEFT ;
      15 ?CaseKeyDef   :: TakeOver SCROLLDOWN  ;
      16 ?CaseKeyDef   :: TakeOver SCROLLRIGHT ;
      47 ?CaseKeyDef   :: TakeOver  TRUE ' LAM EXIT STO ;
        3 #=casedrpfls   DROP
      'DoBadKeyT
     ;
      2DROP 'DoBadKeyT
  ;
    FALSE FALSE
    NULL{}
    ONEFALSE
    ' LAM EXIT
    ' ERRJMP
    ParOuterLoop
    RECLAIMDISP
;
@
```

FILER (Solo ROM 1.19-6)
Lo que veremos a continuación solo funciona en la ROM 1.19-6, ha sido probado en un emulador con la ROM 1.18 y no funciona en esta ROM. Tal vez funcione en ROMs superiores a la 1.18. Pruebe en ROMSs inferiores a la 1.19-6 bajo su propio riesgo.

Nosotros podemos crear un programa para personalizar el FILER de la HP49G.

SINTAXIS:

```
::
 { }                                  %Tipos de Archivos
 { }                                  %Directorio inicial
   {
    { "Etiqueta 1"  TIPOINICIO ACCIONMENU PROGRAMA ASIGNACION  }
    { "Etiqueta 2"  TIPOINICIO ACCIONMENU PROGRAMA ASIGNACION  }
                                  .
                                  .
                                  .
    { "Etiqueta n"  TIPOINICIO ACCIONMENU PROGRAMA ASIGNACION  }
  }
 ^FILER_MANAGERTYPE
;
```

TIPOS DE ARCHIVOS
Se indica los tipos de archivos que se quiere que muestre el FILER por ejemplo si solo se quiere mostrar grob se usara # 2B1E. Estos tipos tendrán que ir en un lista { }.
A continuación se muestran algunos tipos de archivos.

TIPOS DE ARCHIVOS	DESCRIPCION
# 2933	Números Reales (%)
# 2A2C	Strings ($)
# 2A74	Listas ({})
# 2E48	Nombres Globales (ID)
# 2D9D	Programas (PRG)
# 2AB8	Objetos Algebraicos (ALG)
# 2B1E	Objetos Graficos (GROB)
# 2A96	Directorios (DIR)
# 2B40	Librerias (LIB)
# 2614	Numeros Enteros (ZINT)

DIRECTORIO INICIAL
Indica la fila en donde se desea que empiece el **FILER**.

EJEMPLOS:	**INICIA EN:**
{}	HOME
{ FOTOS.DIR}	HOME/FOTOS.DIR
:n:{}	Inicia en el Puerto n.
:2: {GAMES.DIR}	Inicia en el Puerto 2 en el directorio GAMES.DIR

ETIQUETA
El nombre que tendrá el menú entre “ ” , o un grob de menú (21 x 8), si se desea introducir un menú en blanco entonces **{ NULL$ }.**

TIPOINICIO

Puede ser un sistema binario o un programa que cuando se evalué devuelva un binario del sistema. Este binario permite controlar cuando el programa puede empezarse y puede tener cinco valores diferentes, estos valores se muestran en la siguiente tabla:

TIPOINICIO		DESCRIPCION
ZERO	0	Se puede iniciar el programa en cualquier parte (VAR, PORT, BACKUP, LIB).
ONE	1	El programa puede ser iniciado solo si se esta en un Directorio. (HOME, etc.)
TWO	2	El programa no puede iniciarse si se esta examinando una librería.
THREE	3	El programa no puede iniciarse si se esta examinado un backup en algún puerto.
FOUR	4	El programa sólo puede ejecutarse en la raíz de un puerto (0, 1 o 2).

ACCIONMENU

Pueda ser un sistema binario o un programa que cuando se evalué devuelva un binario del sistema. Este definirá el programa que se ejecutara, estos tipos se muestran en la siguiente tabla:

DESCRIPCION	ACCIONMENU	
Bip	ZERO	0
Información	ONE	1
Hexa	TWO	2
Ver	THREE	3
Arbo	FOUR	4
Up	FIVE	5
MaxUp	SIX	6
Down	SEVEN	7
MaxDown	EIGHT	8
Seleccionar	NINE	9
Updir	TEN	10
DownDir	ELEVEN	11
Menú Previo	TWELVE	12
Menú Siguiente	THIRTEEN	13
Evaluar	FOURTEEN	14
SwapHeader	FIFTEEN	15
Detalles	TWENTYFOUR	24
Editar	TWENTYFIVE	25
Copiar	TWENTYSIX	26
Mover	TWENTYSEVEN	27
Rcl	TWENTYEIGHT	28
Eliminar	TWENTYNINE	29
Renombrar	THIRTY	30
CRDIR	THIRTYONE	31
Ordenar	THIRTYTWO	32
Enviar	THIRTYTHREE	33
HALT	THIRTYFOUR	34
Editar	THIRTYFIVE	35
Recibir	THIRTYSIX	36
Salir	THIRTYSEVEN	37
SOLO ROM 1.19-5 Y SUPERIORES		
pageUp	THIRTYEIGHT	38
pageDown	THIRTYNINE	39
NewObject	FORTY	40
Sort	FORTYONE	41

PROGRAMA
Un programa adicional que se ejecutara en un secundario `::  ;`
`TakeOver` si no se ejecutara ningún programa adicional**.**

ASIGNACION
Se puede asignar un programa a una tecla. El formato de la tecla es un sistema binario en la forma:
Pnn

Donde '**P**' es 0 o 1.

P	PLANO
0	Plano Normal
1	Plano 

nn: Sera el codigo de la tecla

nn: Tambien puede ser codigo de la tecla + codigo de cambio.

CODIGO DE CAMBIO	CAMBIO
40	
C0	

Por ejemplo si se quiere asignar un programa a la tecla  .
El codigo de la tecla **I** () es **#9** + el codigo del cambio  que es **#40**. Entonces el codigo final seria la suma de los dos y este seria #049.

049 = (#40 para  + #09 oara )

PRECAUCION: El formato de la tecla debe ser el quinto elemento de la lista.

```
{ "Enviar" BINT2  BINT33  TakeOver  # 054 }
```

Esto asignara el programa a  .

```
Ejemplo:
::
 { # 2D9D }
 NULL{}
  {
    { "Enviar" BINT2  BINT33  TakeOver  # 118 }
    { "Recibir" BINT2  BINT36  }
    { NULL$ }
    { NULL$ }
    { NULL$ }
    {"SALIR"   BINT0   BINT37  }
  }
  ^FILER_MANAGERTYPE
 ;
@
```

Si no se quiere usar el tipo de archivos entonces se puedes usar **^FILER_MANAGER** en lugar de **^FILER_MANAGERTYPE.**

Ejemplo:

```
::
NULL{}
  {
    { "Enviar” BINT2  BINT33  TakeOver  # 118 }
    { "Recibir” BINT2  BINT36  }
    { NULL$ }
    { NULL$ }
    { NULL$ }
    {"SALIR"   BINT0   BINT37  }
  }
  ^FILER_MANAGER
 ;
@
```

32 EDITOR DE LA HP49G

La HP49G tiene un editor que es mucho mas rapido y mejor que el editor de la HP48. Sin embargo es un editor de uso general y seria util agregar algunas funciones espesificas sin tener que escribir un nuevo editor. La HP49 contiene entradas en la Rom que sirven para manipular el editor. Algunas nos estan bien explicadas pero algo se entienden.

A continuación se veran algunas de ellas, este documento a sido extraido del original de ***Carsten Dominik (dominik@astro.uva.nl)***

EditLine
El string que está revisándose actualmente. También llamado " Buffer" y "línea de comandos". En los diagramas de la pila se usaré **$buf** para él

Posición del cursor
Posición del cursor en el Editline. Un binario. En diagramas de la pila escritos como #cpos.

línea actual
La línea actual en el editor, es decir el substring después del NEWLINE antes del cursor al próximo carácter de NEWLINE.

El editor de pantalla
Es cuando el texto esta revisándose a lo largo y/o ancho, la pantalla de la HP49G muestra sólo una parte del texto. Cuando el cursor se mueve, la ventana debe re-posicionarse para mostrar la nueva posición.

Seleccion
Es una region seleccionada en el buffer la seleecion del substring es mostrado como $sel en los diagrammas de la pila.

Caracteres Invisibles
La HP49G puede mostrar el texto en diferentes fuentes y estilos. Para cambiar entre los conjuntos de fuentes y estilos, se insertan marcadores especiales en el texto que indica un cambio en un conjunto de fuetes o estilos. Estas sucesiones del 3-carácter no son visibles, pero ellos están en la longitud del string y en la posición del cursor.

EJEMPLOS:

a) Seleccione la línea actual y copíelo hacia el portapapeles.

```
::
 TakeOver
 CMD_END_LINE        (Va al final de la linea)
 RCL_CMD_POS         (llama la posicion)
 CMD_STO_FIN         (Almacena la posicion)
 CMD_DEB_LINE        (Empieza la linea)
 RCL_CMD_POS         (llama la posicion)
 CMD_STO_DEBUT       (Almacena el posicion)
 CMD_COPY            (Copia al portapapeles)
;
@
```

Esto puede hacerse más corto realmente usando el comando **SELECT.LINE.**

Lo siguiente es equivalente al anterior.

```
::
 TakeOver
 SELECT.LINE
 CMD_COPY
;
@
```

b) Inserte ':: ;' en una sola línea y posiciona el cursor entre '::' y ';'.

```
::
 TakeOver
 ":: ;"
 CMD_PLUS
 CMD_BAK
 CMD_BAK
;
@
```

c) Inserte una multi-línea ':: ; ' y posiciona el cursor con la sangría extra en la segunda línea.

```
::
 TakeOver
 "::\0A\0A;"
  CMD_PLUS
  CMD_UP
  SPACE$                (Arregla la entrada a dos espacios extras)
  CMD_PLUS
;
@
```

d) Vaya a la siguiente etiqueta. Las etiquetas son líneas que empiezan con ' * '.

```
::
 TakeOver
 "\0A*"                (Nueva línea seguida de *  )
 FindStrInCmd          (Busca, mientras encuentra la posición del cursor)
 IT
 ::
 DROP
 #1+                   (agrega #1  #empieza a mover encima de NL)
 STO_CURS_POS          (Almacena la nueva posición del cursor)
 ;
DROP
;
@
```

e) Programa que efectúa terminación de nombres en el Editor. Necesita encontrar el fragmento de la palabra antes del cursor.

```
::
 RCL_CMD                  (llama a EditLine)
 RCL_CMD_POS              (posicion actual)
 DUP                      (argumento necesitado por GET.W<-)
 GET.W<-                  (posiciona al inicio de la palabra)
 #1+SWAP                  (prepara argumentos para SUB$)
 SUB$                     (obtiene un substring)
;
@
```

f) Cambie la entrada de la línea actual a #N espacios. #N es un BINT y espera en la pila nivel 1. El comando deja las líneas vacías y las líneas que empiezan con un '*' exclusivamente.

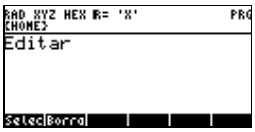
```
::
 Blank$                   (Crea un string en blanco con #N espacios)
 CMD_DEB_LINE             (Empieza la linea)
 RCL_CMD
 RCL_CMD_POS #1+ SUB$1    (Mira el priemr character en la linea)
 BINT42                   (Codigo ASCII de '*')
 OVER#=case               (Si empieza la linea con '*'?)
 ::
   2DROP                  (Borra)
   CMD_DOWN               (Siguiente linea y sale)
 ;
 BINT32 >#?SKIP           (Si las lineas estan con caracteres vacios?)
 ::
   CMD_END_LINE           (Las salidas de las lineas con espacios en blanco:)
   RCL_CMD_POS            (Recuerda el fin de la posicion de la linea)
   CMD_DEB_LINE           (retrocede para empezar la linea)
   DO>Skip                (Salta la siguiente palabra)
   RCL_CMD_POS
   #<ITE
   DROPRDROP              (En la proxima linea: Sale)
   DoFarBS                (Elimina el espacio en blanco antes de la primera palabra)
 ;
 CMD_PLUS                 (Inserta espacios)
 CMD_DEB_LINE             (Retrocede al inicio de la linea)
 CMD_DOWN                 (Siguiente linea)
;
@
```

32.1 EJECUTANDO COMANDOS EXTERNOS EN EL EDITOR

In order to use the new commands in the editor, you must bind them to a key or put them into a menu. Note that each command you write needs a TakeOver as the first call or the command will not execute in the editor.

Here is a simple example for an InputLine environment which defines an initial menu with two commands to select the current line and to clear the EditLine.

```
::
  "Editar"
  ""
  BINT0
  ZEROZEROZERO
  {
    { "Seleccionar"  :: TakeOver SELECT.LINE ; }
    { "Borrar"       :: TakeOver DEL_CMD     ; }
  }
  BINT1
  TRUE
  BINT0
  InputLine
;
@
```



32.2 REFERENCIA

Los siguientes comandos sirven para manipular el editor.

32.2.1 ESTADO

2F196 RCL_CMD
Vuelve a EditLine. (➔ $buf)

2EEEB EDITLINE$
Igual que **RCL_CMD** (➔ $buf)

2F197 RCL_CMD2
Similar a **RCL_CMD**, pero si no hay bastante memoria para copiar EditLine a la pila, pasará el EditLine actual a TEMPOB. Claro, esto anulará el EditLine actual.

2EF87 RCL_CMD_POS
Devuelve la posición del cursor. (➔ #cpos)

26585 CURSOR@
Igual que **RCL_CMD_POS** (➔ #cpos)

32.2.2 INSERTANDO TEXTO

2EF74 CMD_PLUS
Inserta un string en la posicion del curson en EditLine ($ins ➔)

2F194 CMD_PLUS2
Reemplaza EditLine actual con el nuevo string. Cuando no hay bastante memoria para copiar el string en la pila nivel del 1, mueve el string fuera de TEMPOB
El cursor se mueve al extremo del nuevo string ($new ➔)

2F195 CMD_PLUS3
Igual que **CMD_PLUS2**, pero la posición del cursor no se cambia. Útil para restaurar una después de **HALT**. ($new ➔)

2EF97 InsertEcho
Igual que **CMD_PLUS** ($ins ➔)

2F11C Echo$NoChr00
Igual que **CMD_PLUS** ($ins ➔)

32.2.3 BORRANDO TEXTO

2EF82 CMD_DEL
Igul que **LS+DEL.**

2EF81 CMD_DROP
Igual que la tecja **BS,** retrosecede un espacio en el editor.

2EF95 DEL_CMD
Elimina el EDITLINE completo.

2F2F0 DO<Del
Igual que ←**DEL** del menú de editor **TOOL.**

2F2F1 DO>Del
Igual que **DEL**➔ del menú de editor **TOOL.**

2F2F9 DODEL.L
Borra todos los caracteres de la linea actual si la linea ya esta vacia entonces borra la linea.
Igual que la etiqueta '**DEL L**' en el editor del menu **TOOL**

2F2DD DoFarBS
Borra a l empezar la linea.
Igual que **RS**+'←**DEL**' en el editor del menu **TOOL**

2F2DE DoFarDel
Borra al fina de la linea
Igual que **RS+'Del**➔' en el editor del menu **TOOL**

32.2.4 MOVIENDO EL CURSOR

2EF8B STO_CURS_POS
Almacena la posición del cursor. Mueve el cursor a la posición especificada (#pos ➔)

2EF8F STO_CURS_POS_VIS
Como **STO_CURS_POS**, pero ignora los caracteres invisibles.
Así que si usted mira un string y dice, yo quiero ir a lo que yo veo como el 5 carácter, use esta entrada. (#vpos ➔)

2EF7C CMD_NXT
Mueve al siguiente carácter, como la tecla derecha.

2EF7B CMD_BAK
Mueve el cursor a la izquierda. como la tecla izquierda.

2EF80 CMD_DOWN
Mueve a la siguiente línea como la tecla de abajo,

2EF7F CMD_UP
Mueve a la linea previa, como la tecla de arriba

2EF7D CMD_DEB_LINE
Mueve al comienzo de la línea. Igual que **RS+LEFT.**

2EF7E CMD_END_LINE
Mueve al final de la linea. Igual que **RS+RIGHT.**

2EF7A CMD_PAGED
Mueve al final de la pagina, igual que **LS+DOWN.**

2EF77 CMD_PAGEL
Mueve a la pagina izquierda, Como **LS+LEFT.**

2EF78 CMD_PAGER
Mueve a la pagina derecha, Como **LS+RIGHT.**

2EF79 CMD_PAGEU
Sube una pagina, como **LS+UP.**

2F2EE DO<Skip
Igual que ➔**SKIP** del editor del menú **TOOL**

2F2EF DO>Skip
Igual que **SKIP**➔ del editor del menú **TOOL**

2F2E4 DO>BEG
Igual que ➔**BEG** del editor del menú **TOOL**

2F2E5 DO>END
Igual que ➔**END** del editor del menú **TOOL.** Cuando no hay ninguna selección no realiza ningún movimiento.

2F2E6 GOTOLABEL
Lleva arriba los CHOOSE-box con etiquetas en el EditLine.
Igual a la etiqueta del editor del menú **TOOL/GOTO**

32.2.5 SELECCIONAR CORTAR Y PEGAR

2EF83 CMD_STO_DEBUT
Inicia la selección de la marca, como **RS+BEGIN**, pero necesita la posición desde la pila
(#pos ➔)

2EF84 CMD_STO_FIN
Finaliza la marca, como **RS+END**, pero necesita la posición desde la pila (#pos ➔)

2EF85 RCL_CMD_DEB
Almacena la posición del inicio de la marca. Si la selección se ha eliminado devuelve ZERO.
(➔ #pos)
(➔ #0)

2EF86 RCL_CMD_FIN
Almacena la posición al final de la marca. Si la selección de la marca se ha eliminado devuelve ZERO.
(➔ #pos)
(➔ #0)

2F2DC ClearSelection
Borra la selección.

2EF93 VERIF_SELECTION
Devuelve **TRUE** cuando el final de la marca no es **ZERO**, indicando que la selección esta activa. Use este comando como una verificación antes de hacer algo con la selección.
(➔ TRUE)
(➔ FALSE)

2EF8A CMD_COPY
Copia el string seleccionado.

2EF88 CMD_CUT
Corta un string. Realmente lo "borra", no copia para eliminar el buffer. Un "normal" CUT (Cortar) seria así:

```
:: CMD_COPY CMD_CUT ;
```

2EF89 CUT.EXT
Rutina en ML usada por **CMD_CUT**.

2F2FA CMD_COPY.SBR
Coloca la selección del string en la pila. (➔ $sel)

2EF94 PASTE.EXT
Pega desde la Pila con el tratamiento de fuentes y estilos.

2F2E1 SELECT.LINE
Selecciona la actual linea, posiciona le cursor al inicio de la linea.

2F2E2 SELECT.LINEEND
Selecciona la actual linea, posiciona le cursor al final de la linea.

32.2.6 BUSCAR Y REEMPLAZAR

2F2F2 FindStrInCmd
Busca un string el EditLine.
($find ➔ $find $start $end TRUE)
($find ➔ $find FALSE)

2F2F3 GET.W->
Devuelve la posición de la siguiente palabra a la derecha de la posición del actual cursor. (➔ #ws)

2F2F4 GET.W<-
Toma una posición de la pila y devuelve la posición de la siguiente palabra mas cercana a la izquierda del cursor. (#pos ➔ #ws)

2F2E8 DOFIND
Igual a la etiqueta **FIND** del editor del menú **TOOL/SEARCH**. Pops up the FIND input form.

2F2EA DONEXT
Igual a la etiqueta **NEXT** del menú del editor **TOOL/SEARCH**.

2F2E9 DOREPL
Igual a **REP** del menú del editor **TOOL/SEARCH.**

2F2EB DOREPLACE
Igual a **R** del menú del editor **TOOL/SEARCH.**

2F2EC DOREPLACE/NEXT
Igual a **R/N** del menú del editor **TOOL/SEARCH**.

2F2ED REPLACEALL
Igual a **ALL** del menú del editor **TOOL/SEARCH**.

2F2FC REPLACEALLNOSCREEN
Como **REPLACEALL**, pero no actualiza la pantalla. Muy rápido de esta manera.

32.2.7 EVALUACION

2F2DF EditSelect
Edita la selección actual. Abre el editor con la selección. entonces puedes editar la selección.

2F2E3 EVAL.LINE
Evalúa la línea actual y reemplaza con el resultado de la evaluación. Similar a **EVAL.SELECTION**, pero sin la necesidad de seleccionar la línea primero. (? ➔ ?)

2F2FB EVAL.SELECTION
Evalúa la selección actual y reemplaza con el resultado de la evaluación. Igual a **EXEC** del editor del menú **TOOL**. (? ➔ ?)

2F2F8 EXEC_CMD
Ejecute un comando en la selección en el Editline. Toma 2 argumentos : el comando a aplicar y una bandera (flag) que dice cómo compilar la selección antes del comando (cmd T/F ➔ obsel)

32.2.8 MISCELANEA

2EF73 ?Space/Go>
Inserta un espacio a menos que exista ya uno antes de la posición del cursor.

2EF76 AddLeadingSpace
Agrega primero un espacio al string del nivel 1. ($ ➔ $')

2EF75 AddTrailingSpace
Agrega un espacio al string del nivel 1 a menos que el string ya termine con un espacio. ($ ➔ $')

26855 CMDSIZE
Puntero de entrada en ML para conseguir el tamaño del EditLine. Como es una entrada en ML no es posible llamarlo directo desde SystemRPL así que no use esta entrada al menos que sepa de la manera necesaria

```
:: RCL_CMD LEN$ ;
```

Trabajo para los maniacos del Assembler ;-)

2EF9A CommandLineHeight
Devuelve el número de las filas de pixeles ocupadas por la parte visible del EditLine. (➔ #height)

2F2DB DOTEXTINFO
Igual a **INFO** del menú del editor **TOOL**.

2F2F6 GET_CUR_FONT.EXT
Devuelva el ID (como un sistema binario) del conjunto de fuentes usado por el carácter bajo el cursor. (➔ #font)

2EF96 NO_AFFCMD
Cuenta el siguiente **CMD_PLUS** lo llama pero no actualiza la pantalla. Para la velocidad, si usted quiere hacer una inserción antes que el usuario necesita verlo.

2F19E DispCommandLine
Vuelve a actualizar la línea de comandos.

2F19F ?DispCommandLine
Vuelve a actualizar la línea de comandos si es necesario.

2F2F7 PUT_STYLE
Cambia el punto de estilos. Si la selección esta activa, cambia el estilo del texto en la selección. Tomas un BINT de la pila que es el número del estilo. (#style ➔)

Por ejemplo para cambiar al tipo de fuente **ITALI** del menú del editor **TOOL/STYLE** podría llevarse a cabo con el programa siguiente:

```
::
 BINT2
 PUT_STYLE
;
@
```

2F2F5 PUT_FONTE
Cambia los puntos de fuente. Similar al comando **PUT_STYLE.** (#font ➔)

2F2E7 SELECT.FONT
Igual a **FONT** del menú del editor **TOOL/STYLE**.

2F2E0 ViewEditGrob
Ve el grob editado actualmente en el Editline. Si en el EditLine contiene el GROB 10 10 FFFFFFFFFFFFFFFF mueve el cursor al "1" del primer "10". Entonces este punto da la entrada en que se mostrara el grob.

2EF92 XLINE_SIZE
Verifica si el cursor esta fuera de la línea actual. En el editor de la HP49G, usted puede mover el cursor más allá del margen derecho,
Esta entrada vuelve **TRUE** si no esta antes del newline.
(ob ➔ TRUE) ; Fuera de la línea.
(ob ➔ FALSE) ; Dentro de la línea.

HP49 BROWSER

Hay dos comandos del navegador o browser en la calculadora HP49G, el primero que estaba presente desde la serie HP48 y uno nuevo que sólo esta presente en el modelo HP49G. Este capítulo describirá el nuevo comando que es más fácil usar, tiene algunos rasgos que el primero no tiene, pero el primero también tiene algunos rasgos importantes que este nuevo no hace.

Hay varios flashpointers que pueden ser usados para acceder al navegador, estos flashpointers no se apoyan oficialmente pero son muy estables.

La entrada principal es **FPTR 2 72 (ˆChoose3)**. tiene el siguiente diagrama en la pila:
(Meta $Titulo #Init **::**Message ➔ ob TRUE) o
(Meta $Titulo #Init **::**Message ➔ FALSE)
dependiendo adelante si el usuario selecciona algo o lo cancela.
Como una alternativa, usted puede reemplazar **FPTR 2 72** con **FPTR 2 74**.

Meta
Contiene los ítems que deben mostrarse en el choose.
Todos los tipos de objeto se permiten y ellos se descompilan por la pantalla.

$Titulo
Es el titulo que se quiere que figure en la parte superior del choose, este deberá ir dentro de un string, si no se desea ningún titulo entonces se ingresa un string vació.

#Init
Es el ítem que estará seleccionado por defecto, generalmente es el primer ítem.
El entero binario **ZERO** (BINT 0) indica el primer subtitulo.
.
::Message
Message es un programa, esta aplicación llamará al mensaje normalmente un bint en el nivel uno de la pila y quizá los argumentos adicionales en otros niveles de la pila.
::Message debe hacer su trabajo y retornar **TRUE**. Si decide ignorar **::**Message debe dejar caer simplemente el bint y retornar **FALSE**.
Para que funcione **::**Message debe tener por consiguiente el comando **DROPFALSE** (qué se puede poner en la pila con **' DROPFALSE**).
Cuando usted usa el **xCHOOSE** de usuario, apenas proporciona **DROPFALSE**.

::Message puede ocuparse de los siguientes mensajes:

Mensaje Nombre del mensaje y significado

BINT1 MSGDISPBOX
Este mensaje tiene que ver con la pantalla del choose, actualmente no es bien entendido.
El diagrama en la pila para este mensaje parece ser
(#1 ➔ **::**prog TRUE)
(#1 ➔ FALSE)

BINT2 MsgDispTitle
Esto debe desplegar el título. Si no se maneja el título es arrastrado
usando el argumento proporcionado por $title.
(#2 ➔ TRUE)
(#2 ➔ FALSE)

BINT3 MsgEndInit
Este mensaje se ejecuta después de la inicialización del choose antes de que el mando se entregue al POL.
(#3 ➔ TRUE)
(#3 ➔ FALSE)

BINT4 MsgKeyPress
Éste es un negociante importante, similar al ones usado por el POL. Cuando el usuario presiona una tecla, el message handler hace un llamado al código de tecla y plano y el mensaje BINT4 en la pila. Debe devolver una dirección o un secundario, **TRUE** y **TRUE** de nuevo.

Si la tecla no esta ocupada debe devolver **FALSE**.
Este es el diagrama en la pila para el ::message con respecto a este mensaje:
(#kc #pl #4 ➔ KeyDef TRUE TRUE)
Si, TRUE dos veces
(#kc #pl #4 ➔ FALSE)

BINT5 MsgMenu
Esto debe devolver el menú que se muestra al usuario durante la selección. El valor del retorno para este mensaje se evalúa para conseguir el menú. El menú no se actualiza automáticamente cuando usted mueve la selección, pero puede usarse el mensaje #6 para forzar a una actualización.
Si el menú tiene más de una página, usted debe ocuparse del **NXT** y **PREV** codificándolos en el keyhandler para que se puedan manejar por que estos no se manejan por defecto.
(#5 ➔ { menu list } TRUE)
(#5 ➔ ::prog_returning_list TRUE)
(#5 ➔ FALSE)

BINT6 MsgEndEndDisp
Este mensaje se llama después de un redisplay del choose (Cuando uno cambia un subtitulo seleccionado). Se puede usar esto para forzar a una actualización del despliegue del menú poniendo **24LAM** a **FALSE**.
(#6 ➔ TRUE)
(#6 ➔ FALSE)

El Browser y sus Lams
El navegador POL usa 24 variables locales anónimas.
A continuación la descripción de algunos LAMs importantes usados por el navegador:

LAMs	DESCRIPCION
1LAM	Quit. Ponga esto a **TRUE** si usted quiere que el POL termine
2LAM	DispOffset. Indice del ítem seleccionado respecto a DispTop.
3LAM	DispTop. Indice del primer ítem actualmente visible en la pantalla.
14LAM	Programa redisplay.
17LAM	**::**Message
24LAM	DisplayMenu. Ponga esto a **FALSE** para forzar a un redisplay del menú.

ACCEDIENDO AL ITEM SELECCIONADO
Para usar el navegador, para seleccionar más de un ítem, se deben escribir programas que deben ser accesibles con el key handler o con el menú.

En la mayoría de tareas se debe averiguar lo que el ítem actual es, el choose guar da dos copias de la lista del choose en el Virtual Stack (Pila Virtual) y usted puede usar estos para conseguir el ítem actual. En el nivel uno del Virtual Stack la lista es invertida y los ítems que se tienen que mostrar en el choose son convertidos en strings (sensible al flag **-85**). En los tres niveles de la pila virtual hay una copia de la lista original
El índice del ítem actual esta disponible con el siguiente código:

```
::
 2GETLAM
 3GETLAM
 #+
;
@
```

Los índices empiezan en 0.
Para acceder al ítem actual use uno de estos métodos:

1. Consigue el string descompilado. Esto es muy rápido y sólo ocupa 8 bytes.

```
::
 2GETLAM
 3GETLAM
 #+
 GetElemBotVStack
;
@
```

2. Consigue el ítem original.
No hay ninguna manera para conseguir directamente el tercer nivel de la Pila Virtual, por lo cual habrá que excavarlo hacia afuera y restaurar después la pila.
A continuación una manera para hacerlo: (35 bytes):

```
::
 GetVStackProtectWord PopMetaVStack
 GetVStackProtectWord PopMetaVStack
 2GETLAM 3GETLAM #+
 GelElemTopVStack
 1LAMBIND
 PushMetaVStack&Drop SetVStackProtectWord
 PushMetaVStack&Drop SetVStackProtectWord
 1GETABND
;
@
```

Esto parece complicado, pero también es bastante rápido y realmente esto es usado por la ROM por la tecla de ayuda del catálogo.

3. Si usted busca algo demasiado largo, usted puede guardar una copia de su lista original, por ejemplo en un LAM nombrado "mylist". Si usted llama al flashpointer del navegador, usted consigue el elemento actual con:

```
::
 LAM mylist
 2GETLAM
 3GETLAM
 #+ #1+
 NTHCOMPDROP
;
@
```

GUARDANDO Y RESTAURANDO LA PANTALLA
Si usted quiere usar una tecla del menú u otra tecla para hacer una excursión del choose de la pantalla, usted debe guardar y debe restaurar la pantalla actual.
Hay dos flashpointer que pueden usarse para guardar y restaurar la pantalla:

FPTR 2 88 Guarda la pantalla actual
FPTR 2 89 Restaura la pantalla guardada.
Si un choose crea otro choose que necesita guardar su pantalla para una excursión
En casos así usted necesita guardar y restaurar copias de HARDBUFF y HARDBUFF2.

A continuación algunos comandos de referencia.

072002 ^Choose3
(meta $titulo #pos ::handler ➔ ob T)
(meta $titulo #pos ::handler ➔ F)

074002 ^Choose3Index
Igual que **^Choose3**, pero retorna el índice del ítem seleccionado el lugar del propio ítem.
El índice empieza en 0.
(meta $titulo #pos ::handler ➔ #idx T)
(meta $titulo #pos ::handler ➔ F)

070002 ^Choose2
Llama a **^Choose3Index** con el **::**Message vacio.
Esto es simplemente
```
:: 'DROPFALSE FPTR2 ^Choose3Index ;
```
(meta $titulo #pos ➔ ob T)
(meta $titulo #pos ➔ F)

073002 ^Choose3Save
(meta $titulo #pos ::handler ➔ ob T)
(meta $titulo #pos ::handler ➔ F)
Guarda y restaura HARDBUFF y HARDBUFF2 alrededor de una llamada por Choose3.

005002 ^sysCHOOSE
($titulo {} %sel ➔ ob %1)
($titulo {} %sel ➔ %0)
Equivalente al comando CHOOSE de User RPL.

075002 ^ChooseDefHandler
Avanza al mensaje por defecto (el usado por la tecla CAT) en la pila.
(➔ **::**Message)

088002 ^SaveHARDBUFF
Guarda HARDBUFF y HARDBUFF2 en un lugar seguro.
(➔)

089002 ^RestoreHARDBUFF
Restaura HARDBUFF y HARDBUFF2 si fueron guardados con SaveHARDBUFF.
(➔)

077002 ^Choose3OK
La acción **OK** ejecutada por Choose3 si **OK** o **ENTER** se presiona.
(➔)

076002 ˆChoose3CANCL
La acción **CANCL** ejecutada por Choose3 si **CANCL** o **ON** se presiona.
(➔)

Ejemplo:
Este programa despliega los números del 1-100 si se presiona Squareroot entonces saca la raíz cuadrada del número actual en un cuadro de mensaje.
En el menú **F1** agrega los números seleccionados a una lista cada número en un string,
F3 no hace nada pero muestra si el número seleccionado es par o impar, **F5** y **F6** son los usuales **CANCL** y **OK.**

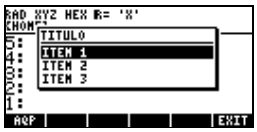
```
::
  101 ONE_DO                          (PARAMETRO Meta)
  INDEX@
  UNCOERCE
  LOOP
  100 P{}N                            (Crea una lista del 1 al 100)
  DUP
  NULL{}                              (Lista de colección vacia)
  { LAM mylist  LAM res } BIND        (Guarda y copia en la lista)
  INNERCOMP
  "REALS"                             (PARAMETRO  $Titulo)
  0                                   (PARAMETRO  #Init)
  '
  ::                                   (PARAMETRO ::Message)
    4
    OVER#=case
    ::
      DROP DUP#1=
      3PICK
      23 #=
      ANDcase                         (Si se presiona SQRT)
      ::
        2DROP                         (Borra los códigos de teclas)
        '
        ::
          LAM mylist
          2GETLAM
          3GETLAM
          #+  #1+
          NTHCOMPDROP                 (Llama al valor actual)
          %SQRT DO>STR                (Calcula la raíz)
          FlashWarning                (Muestra la raíz)
        ;
        TrueTrue                      (Si manejamos estas teclas)
      ;
      FALSE                           (Las otras teclas no se manejan)
    ;
    5
    OVER#=case                        (Menú)
    ::
      DROP
      '
      ::
        NoExitAction                  (No guarda el ultimo menú)
        {
          {
            "->{}"
            ::
```

```
            TakeOver
            LAM res
            2GETLAM  3GETLAM
            #+
            GetElemBotVStack
            >TCOMP
            '
            LAM res
            STO
          ;
        }
        NullMenuKey
        {
          ::                        (Menú a mostrar)
            TakeOver                ("par" o "impar")
            LAM mylist              (la lista)
            2GETLAM
            3GETLAM #+ #1+          (Consigue el elemento actual)
            NTHCOMPDROP
            DUP %2 %/               (Prueba si es par)
            %FLOOR %2 %* %=
            ITE                     (Retorna la etiqueta correcta)
            "par"
            "impar"
          ;
          NOP                       (Ninguna acción cuando se aprieta)
        }
        NullMenuKey
        { "CANCL"  FPTR 2 77 }
        { "OK" FPTR 2 76 }
      }
    ;
    TRUE                            (Si se proporciona un menú)
  ;
  6                                 (Fuerza a la actualización del menú)
  OVER#=case
  ::
    DROP
    FalseFalse
    24 PUTLAM
  ;
  DROPFALSE                         (Otros  mensajes)
;
FPTR 2 72                           (Inicia  el choose)
ITE
::
  DROP                              (Borra el valor actual)
  LAM res                           (Retorna la lista)
  TRUE
;
FALSE                               (Si se presiona CANCL retorna FALSE)
ABND                                (Abandona el entorno temporal)
;
@
```

HP48 BROWSER

El navegador de la HP48 qué todavía está presente en la HP49 permite hacer muchas cosas, básicamente despliega una lista de ítems en donde se puede seleccionar uno o mas ítems y se puede actuar en esas entradas por medio de las teclas del menú o las asignaciones de las teclas duras (hard keys).

full-screen



windowed

Este navegador tiene algunos rasgos que el nuevo navegador de la HP49 no tiene, como el modo full-screen , sin embargo es más complicado de usar. Simplemente como el Input Form hay generalmente varias maneras de lograr la misma cosa.

El navegador se llama por la entrada **˜Choose,** si se usa una HP49 con ROM1.18 se llama con la entrada `ROMPTR B3 0`. Espera cinco parámetros en la pila, devuelve los resultados y **TRUE** o simplemente **FALSE**.
Aquí está el diagrama de la pila:
(::Appl $Titulo ::Converter {}Items Init ➔ result TRUE) o
(::Appl $Titulo ::Converter {}Items Init ➔ FALSE)
Aquí el resultado es una lista o un solo objeto, dependiendo si se habilitan las marcas del check y las selecciones múltiples.

::Appl
Éste es un programa que permite la configuración de varios aspectos del navegador. Se llama con un bint en la pila, que representa el código del mensaje.
Si el mensaje se ocupa, el programa debe devolver cualquier dato requerido por el mensaje y **TRUE** de lo contrario devuelve **FALSE**.
Para que funcione **::Appl** debe tener por consiguiente el comando **DROPFALSE** (qué se puede poner en la pila con el comando **' DROPFALSE**).

A continuación las descripciones de algunos de los mensajes:

Codigo (Decimal) Description y Stack

57 Número de líneas que el navegador desplegará en la pantalla. El valor por defecto depende del conjunto de caracteres actual y en la bandera del sistema 90.
(➔ #)

58 Altura de la línea del navegador. Probablemente esto no necesita ser cambiado.
(➔ #

59 Ancho de la línea del navegador. Deja el espacio para el despliegue de las flechas en la pantalla si el número de elementos es mayor al tamaño de la página
(➔ #)

60 Deberá ser **TRUE** si el navegador estará en el modo full-screen o **FALSE** si estará en el modo windowed. El valor por defecto es windowed.
(➔ flag)

61 Debe ser **TRUE** si se permiten las marcas del check, así se puede apoyar la selección de varios ítems, o **FALSE** si no se quiere. Por defecto no se permiten las marcas del check, ósea no se pueden seleccionar varios ítems. .
(➔ flag)

62 Retorna el número de elementos. Si su programa cambia el número de elementos durante la ejecución, usted debe ocuparse de este mensaje.
(➔ #)

63 Debe retornar las coordenadas de la esquina izquierda superior de la selección de navegador. Generalmente no se necesita cambiar el valor por defecto.
(➔ #x #y)

64 Este mensaje debe devolver la diferencia inicial entre el ítem seleccionado y la cima de página. Esté seguro que la diferencia está menos de la selección actual y por otra parte menos del tamaño de la página, la calculadora puede chocar.
(➔ #)

65 Este mensaje se llama cuando el fondo necesita ser pintado. Su acción puede usarse para dibujar algo más en el fondo.
(➔)

66 Este mensaje se llama cuando el título necesita ser pintado. Su acción debe dibujar el título en HARDBUFF. La mayoría de las veces, esto no se maneja, y el título es arrastrado del parámetro de $Titulo.
(➔)

67 Retorna el título como un grob. La mayoría de las veces esto no se maneja y el título es arrastrado del parámetro de $Titulo.
(➔ grob)

68 Si el mensaje **67** no se define, este se llama para devolver el título como un grob, pero sólo para el modo full-screen.
(➔grob)

69 Igual que el mensaje **68** pero solo para el modo windowed.
(➔grob)

70 Si el parámetro $Titulo no es un string nulo, esta entrada se llama para devolver el título del string. Esto atropella el parámetro $Titulo.
(➔ $)

74 Este mensaje dibuja todas las líneas visibles del navegador.
(➔)

79 Este mensaje debe mostrar una línea del navegador. Si ésta es la línea seleccionada, este mensaje debe dibujar esta línea inversa a la marca de uno que este seleccionado de otra manera.
(# ➔)

80 Este mensaje es una alternativa al parámetro {}Items. Proporciona el número de ítems y este mensaje debe retornar el ítem. Puede retornar cualquier objeto.
::Converter llamará para convertir esto en un string.
Si usted quiere cambiar o variar los ítems del navegador este mensaje permite hacer eso. Pero el mensaje **82** probablemente es mejor en este caso.
(# ➔ objeto)

81 Este mensaje convierte un elemento en un grob. (Esto atropella el parámetro ::Converter.) debe devolver un grob con dimensiones 7NULLLAMx8NULLLAM.
Si se habilitan las marcas del check, usted debe incorporar la marca del check en el grob si el artículo se verifica.
(# ➔ grob)

82 Este mensaje es como el mensaje **80**, pero el objeto ya se retorna como un string.
::Converter ya no sera llamado después. Si se usa este mensaje ya no se necesita escribir el parámetro ::Converter.
(# ➔ $)

83 Retorna una lista que describe el menú. El formato de la lista es el mismo de InputLine y Input Forms,
(➔ {})

85 Este mensaje se llama cuando se inicia el navegador.
(➔)

86 Este mensaje se llama cuando un item se verifica o no se verifica (check). La acción por defecto se ocupa de la verificación y no verificación de los items, probablemente no sea necesario ocuparse de este mensaje.
(# ➔)

87 Este mensaje se llama antes de salir del navegador.
(➔)

91 Este mensaje se llama después de presionar la tecla **ON** o la tecla del menú **CANCL**.
Si retorna **TRUE** sale del navegador, si retorna **FALSE** continua con el navegador.
(➔flag)

96 Este mensaje se llama después de presionar la tecla **ENTER** o la tecla del menú **OK**.
Si retorna **TRUE** sale del navegador, si retorna **FALSE** continua con el navegador.
(➔flag)

$Title
Este parámetro especifica el título. Hay mensajes que pueden atropellar este parámetro:
66, **67**, **68**, **69** y **70**.

::Converter
Este es un secundario que convierte cualquier objeto que es usado en una lisa en un string para ser mostrado. El diagrama de la pila para este secundario es:
(objeto ➔ $)

Si uno se ocupa del mensaje **81** o **82**, no es necesario escribir este programa para hacer la conversión. Sin embargo el navegador le permite al usuario presionar **ALPHA** seguido por una letra para buscar un objeto que empiece con esa letra para saltar a dicho objeto. Esto requiere del parámetro ::Converter aun cuando estos mensajes ya sean proporcionados. Así que uno se debe asegurar que este parámetro devuelva un string.
El comando **DO>STR** puede ser de gran utilidad aquí.

{}Items
Usted puede especificar una lista de objetos aquí o se puede especificar una lista vacía y se puede usar los mensajes **80**, **81** o **82** para proporcionar los elementos.

Init
Éste puede ser un entero binario o una lista.
Si es el **BINT0**, el navegador trabaja como un espectador desaprobando las selecciones.

Si es cualquier otro BINT es el elemento inicialmente seleccionado como por ejemplo el **BINT1** indica el primer item seleccionado.

Si se habilitan las selecciones múltiples, usted puede especificar una lista de bints representando los elementos inicialmente verificados.

USO TIPICO DEL NAVEGADOR
Como se a notado existen varias maneras de proporcionar los elementos que formaran la lista del navegador, seguramente usted puede estar un poco desconcertado por eso aquí se listarán dos maneras de hacer esto.

1) Puede proporcionar los elementos que se usaran en el parámetro Items y proporcionar el parámetro ::Converter que convertirá uno de esos elementos en un string. En este caso no se necesita preocuparse por los mensajes **80**, **81** o **82**. Este método es bueno si la lista de elementos no cambiará mientras el programa está corriendo.

2) Puede proveer una lista vacía al parámetro {}Items, y guardar la lista de elementos en alguna otra parte (probablemente en un lam). Entonces se usaran los mensajes **80**, **81** o **82** para devolver los elementos.

Si se usa los mensajes **81** o **82**, usted ya devolverá los elementos como un grob o un string y el parámetro ::Converter puede ser un secundario nulo, también usted puede usar el mensaje **80** para devolver algún objeto y entonces usar el parámetro ::Converter para extender un string. Este método es bueno si los elementos cambian mientras el programa está corriendo. Si se usa esta técnica también se debe ocupar del mensaje **62**.

Cuando cambia el número de elementos del navegador ejecute este código para adaptar el navegador a los cambios:

```
::
 ROMPTR 0B3 03E   (Vuelve a leer el # de elementos)
 ROMPTR 0B3 026   (Vuelve a leer el ancho)
 18GETLAM         (# Indice)
 5 12GETLAM       (# de Elementos)
 DUP#0=IT
 DROPONE
 #MIN             (Reduce # indice si el # de Elementos es reducido)
 10 18PUTLAM
 FALSE ROMPTR 0B3 019 (Recalcula el desplazamiento)
;
@
```

A continuación algunos comandos como referencia.

0000B3 ˜Choose
(::Appl $Titulo ::Convert {} offset ➔ {}' T)
(::Appl $Titulo ::Convert {} offset ➔ ob T)
(::Appl $Titulo ::Convert {} offset ➔ F)
Retorna los valores en una lista si se habilitan los campos del check, De lo contrario simplemente retorna el objeto seleccionado. Solo retorna **FALSE** cuando el usuario presiona **CANCL**.

0050B3 ˜ChooseMenu0
Devuelve un menú con **"OK"**.
(➔ {})

0060B3 ˜ChooseMenu1
Devuelve menús con **"CANCL"**, **"OK"**.
(➔ {})

0070B3 ˜ChooseMenu2
Devuelve menús con "**CHK"**, **"CANCL"**, **"OK"**.
(➔{})

0630B3 ˜ChooseSimple
Interfase del choose simple de la HP48. En la HP49G llama a ˆRunChooseSimple.
($titulo {ítems} ➔ ob T)
($titulo {ítems} ➔ F)

004002 ˆRunChooseSimple
Interfase del choose simple de la HP48.
($titulo {ítems} ➔ ob T)
($titulo {ítems} ➔ F)

09F002 ˆDoCKeyCheck
Marca el ítem actual.
(➔)

0A0002 ˆDoCKeyChAll
Marca *(Check)* todos los ítems.
(➔)

0B0002 ˆDoCKeyUnChAll
Desmarca *(Uncheck)* todos los ítems.
(➔)

09E002 ˆDoCKeyCancel
Simula Cancel.
(➔)

09D002 ˆDoCKeyOK
Simula OK.
(➔)

0B3002 ˆLEDispPrompt
Redibuja el titulo.
(➔)

0B2002 ^LEDispList
Redibuja las líneas del navegador.
(➔)

0B1002 ^LEDispItem
Redibuja una línea.
(# ➔)

0150B3 ~BBMoveTo
Mueve la línea seleccionada y actualiza la pantalla.
(# ➔)

0190B3 ~BBRecalOff&Disp
Recalculala el desplazamiento del ítem seleccionado en la pagina y vuelve a dibujar las líneas si el flag es **TRUE.**
(flag ➔)

0220B3 ~BBRunEntryProc
Envia el mensaje **85** a **::Appl**.
(➔)

0230B3 ~BBReReadPageSize
Re-lee el tamaño de la pagina (mensaje **57**).
(➔)

0240B3 ~BBReReadHeight)
Re-lee la altura de la línea del navegador (mensaje **58**).
(➔)

0250B3 ~BBReReadCoords)
Re-lee las coordenadas del cuadro del navegador. (mensaje **63**).
(➔)

0260B3 ~BBReReadWidth
Re-lee el ancho de la línea del navegador (mensaje **59**).
(➔)

0280B3 ~BBRunENTERAction
Envía el mensaje **96** a **::Appl**, ejecutando así la acción **OK**. No verifica el valor devuelto y nunca sale.
(➔)

0290B3 ~BBRunCanclAction
Envía el mensaje **91** a **::Appl**, ejecutando así la acción **CANCL**. No verifica el valor devuelto y nunca sale.
(➔)

02F0B3 ~BBReDrawBackgr
Vuelve a dibujar el fondo.
(➔)

0370B3 ~BBGetNGrob
Retorna el número de elementos como un grob.
(#n ➔ grob)

0380B3 ˜BBGetNStr
Retorna el número de elementos como un string.
(#n ➔ $)

03B0B3 ˜BBRereadChkEnbl
Re-lee si están habilitadas las marcas del Check. (Mensaje **61**).
(➔)

03C0B3 ˜BBRereadFullScr
Re-lee si esta en el modo full-screen. (Mensaje **60**).
(➔)

03D0B3 ˜BReReadMenus
Re-lee el menú. (Mensaje **83**).
(➔)

03E0B3 ˜BBReReadNElems
Re-lee el número de elementos. (Mensaje **62**).
(➔)

03F0B3 ˜BBGetN
Retorna el número de elementos.
(#n ➔ ob)

04B0B3 ˜BBIsChecked?
Retorna el número de elementos verificados
(#n ➔ flag)

0520B3 ˜BBUpArrow
Retorna la flecha arriba como un grob.
(➔grob)

0530B3 ˜BBDownArrow
Retorna la flecha abajo como un grob.
(➔grob)

0540B3 ˜BBSpace
Retorna un espacio como un grob.
(➔ grob)

0590B3 ˜BBPgDown
Baja una pagina.
(➔)

05A0B3 ˜BBPgUp
Sube una página.
(➔)

05B0B3 ˜BBEmpty?
Retorna **TRUE** si el navegador no tiene elementos.
(➔flag)

05C0B3 ˜BBGetDefltHeight
Retorna la altura de las líneas basado en el tipo de fuente usada.
Este valor es la altura predefinida por el navegador. Equivalente a **FPTR 2 64.**
(➔#)

0190E0 ˜BRRclC1
```
:: LAM 'BR5 ;
```
(➔)

NULLLAMs Usadas por el Navegador
El navegador usa un gran número de NULLLAMs *(Variables temporales sin nombre)* para guardar su información. A continuación una breve descripción de estas variables:

LAMS	DESCRIPCION	TIPO
1	Usado por **CACHE.**	n/a
2	Condicion de salida de **POL.**	Flan
3	Estado inicial de la pantalla. Es una lista en el formato: `{ DA1IsStatFlag DA2bEditFlag DA1BadFlag DA2aBadFlag DA2bBadFlag DA3BadFlag }`	{}
4	Menú antes de que el navegador empiece.	grob 131x8
5	Pantalla antes de que el navegador empiece.	grob 131x56
6	Desplazamiento en la pagina.	#
7	Altura de la linea del navegador	#
9	Coord. **X** de la esq. Izq. superior del navegador en el HARDBUFF	#
10	Coord. **Y** de la esq. Izq. superior del navegador en el HARDBUFF	#
11	Tamaño de la pagina	#
12	Numero de elementos	#
13	Menú	{}
14	Modo Full-Screen?	flag
15	Lista de indices de flan de los ítems verificados (Check)	flag
16	Marcas del Check habilitadas?	flag
17	**TRUE** si es un navegador o **FALSE** si es un espectador	flag
18	Indice actual seleccionado	#
19	{}Ítems	{}
20	**::**Converter	**::**
21	$Title	$
22	**::**Appl	**::**

Ejemplo:
Este ejemplo usa el navegador para permitirle al usuario entrar en una lista de ecuaciones
Inicialmente la lista está vacía. El usuario agrega las ecuaciones a la lista.
También pueden revisarse las ecuaciones o anuló.
Este programa se ocupa de los mensajes **62** y **82** para devolver el número de elementos
y la ecuación ya convertida en un string cuando esta se pida.

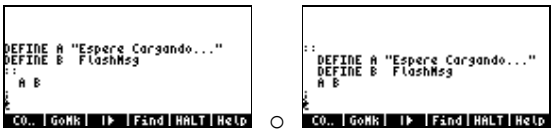
```
::
  NULL{}                              (Emepezar con una lista vacia)
  '
  LAM EQS
  1 DOBIND
  '                                   (PARAMETRO ::Appl)
  ::
    60                                (Modo Full-Screen)
    #=casedrop
    TrueTrue
    62                                (Número de elementos)
    #=casedrop
    ::
      LAM EQS
      LENCOMP DUP#0=IT
      #1+ TRUE
    ;
    82                                (Retornar # elementos en un string)
    #=casedrop
    ::
      LAM EQS
      SWAP NTHELCOMP
      ITE
      ::                              (Convierte en un string)
        setStdWid
        FPTR 4 7
      ;
      "No ecuaciones"  TRUE
    ;
    83                                (Menú)
    #=casedrop
    ::
      {
        {
          "Add"
          ::
            PushVStack&Clear          (Guarda el stack)
            DoNewEqw
            DEPTH #0<>
            IT
            ::                        (Agrega una ecuación)
              LAM EQS
              SWAP >TCOMP
              '
              LAM EQS
              STO
              ROMPTR B3 3E            (Vuelve a leer el # elementos)
            ;
            PopMetaVStackDROP
        ;
        }
        {
```

```
        "Del"
        ::
          LAM EQS
          INNERDUP
          #0=case DROP              (Sale si está vacío)
          PushVStack&Keep           (Guarda el contenido del stack)
          reversym  DROP 18GETLAM
          ROLL DROP 18GETLAM  #1-
          UNROLL DEPTH {}N
          '
          LAM EQS
          STO
          PopMetaVStackDROP         (Restaura el stack)
          ROMPTR B3 3E               (Vuelve a leer el # elementos)
          18GETLAM                  (Cambia el elemento seleccionado)
          12GETLAM                  (Si es necesario)
          #MIN 18PUTLAM FALSE
          ROMPTR B3 19
        ;
      }
      {
        "Edit"
        ::
          LAM EQS                   (llama a los elemetos)
          18GETLAM  NTHELCOMP
          NOT?SEMI                  (Sale si esta vacio)
          FPTR2 ^EQW3Edit           (Edita)
          75 NOT?SEMI               (Sale si no cambia)
          18GETLAM LAM EQS
          PUTLIST                   (Reemplaza)
          '
          LAM EQS
          STO
        ;
      }
      NullMenuKey
      {
        "CANCL"  FPTR 2 9E
      }
      {
        "OK" FPTR 2 9D
      }
    }
    TRUE
  ;
  DROPFALSE
;
"HP48-BROWSER"                      (PARAMETRO ::Titulo)
' NULL::                            (PARAMETRO ::Converter)
NULL{}                              (PARAMETRO ::{ }Items)  (No items)
1                                   (PARAMETRO ::Init)
~Choose                             (Empezar navegador)
ABND                                (Abandonar entorno temporal)
;
@
```

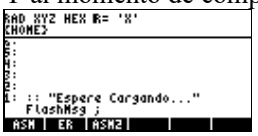
DEFINE, INCLOB, INCLUDE

Estas palabra sirven para facilitar la compilación veamos cada uno de ellos.

DEFINE.
Este nos sirve para definir algún objeto como por ejemplo supongamos que se va usar al momento de hacer un programa el string “Espere Cargando…” y el comando FlashMsg.
Ejemplo:



Y al momento de compilar el programa nos queda:

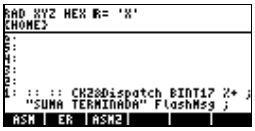


Nota: Todo lo que se quiera definir se tendra que poner en la misma fila de DEFINE por ejemplo si se quiere definir **2 3 4 5** se puede hacer **DEFINE aqp 2 3 4 5** Esto quiere decir que el cualquier parte del programa donde coloquemos la palabra **aqp** al momento de compilar se va a remplazar por **2 3 4 5**.

INCLOB
Esto sirve para incluir el contenido de un archivo en el punto donde se encuentre INCLOB.
Por ejemplo supongamos que tenemos la variable de nombre SUMA que contiene el programa

Si hacemos un programa como se muestra en la figura.

Al momento de ensamblar nos queda:

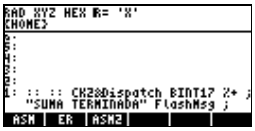


INCLUDE

Esto sirve para incluir el código fuente de un programa en el punto donde se encuentre INCLUDE. Por ejemplo supongamos que tenemos la variable de nombre SUMA que contiene el código fuente:

Si hacemos un programa como se muestra en la figura.

Al momento de ensamblar nos queda:



A continuación se verán algunos ejemplos:

EJEMPLO 1: HP49 IF/THEN/END

```
USER-RPL
«
 IF
 1 >
 THEN
 "MAYOR QUE 1"
 END
 1 DISP 1 FREEZE
»

SYS-RPL
::
 %1
 %>
 IT
 ::
  "MAYOR QUE 1"
   DISPROW1
   SetDA1Temp
 ;
;
@
```

EJEMPLO 2: HP49 IF/THEN/ELSE/END

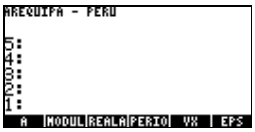
```
USER-RPL
«
 IF
 1 >
 THEN
 "MAYOR QUE 1"
 ELSE
 "MENOR que 1"
 END
 1 DISP 1 FREEZE
»

SYS-RPL
::
 %1
 %>
 ITE
 "MAYOR QUE 1"
 "MENOR QUE 1"
 DISPROW1
 SetDA1Temp
;
@
```

EJEMPLO 3: HP49 Mostrar Texto

SYS-RPL

```
::
 CLEARLCD
 ZEROZERO
 "AREQUIPA - PERU"
 $>grob
 XYGROBDISP
 SetDA1Temp
;
@
```



EJEMPLO 4: HP49 SUMA

USER-RPL

```
«
 2 5 +
 "123" 1 DISP
»
```

SYS-RPL

```
::
 %2
 %5
 %+
 "123"
 DISPROW1
;
```

EJEMPLO 5: HP49 Variables temporales

USER-RPL

```
« → a b c « c b a » »
```

SYS-RPL

```
::
 CK3
 { LAM a
   LAM b
   LAM c
 }
 BIND
 LAM c
 LAM b
 LAM a
 ABND
;
@
```

EJEMPLO 6: HP49 BEGIN LOOP

En el siguiente ejemplo se vera una línea que se desplaza por la pantalla.

```
SYS-RPL
::
 BEGIN
 131 0
 DO
 INDEX@
 0
 OVER
 16
 TOGLINE
 LOOP
 ATTN?
 UNTIL
;
@
```

EJEMPLO 7: HP49 Mostrar un grob

A continuación se vera un ejemplo de cómo mostrar un grob en la pila. Para este ejemplo vamos a suponer el grob esta almacenado en un archivo de nombre FOTO.

```
::
 ZEROZERO
 ID FOTO
 XYGROBDISP
 SetDAsTemp
;
@
```

EJEMPLO 8: HP49 DoinputForm

```
::
  "SysRpl Stack"  BINT8  BINT10
  "MASD SysRpl"   BINT9  BINT19
  'DROPFALSE
  BINT0  BINT7
  BINT6  BINT9
  BINT32
  MINUSONE  MINUSONE
  ""
  MINUSONE  MINUSONE
  TRUE
  :: BINT85  TestSysFlag  ;
  'DROPFALSE
  BINT0  BINT16
  BINT6  BINT9
  BINT32
  MINUSONE  MINUSONE
  ""
  MINUSONE  MINUSONE
  TRUE
  :: BINT92  TestSysFlag  ;
  BINT2  BINT2
  'DROPFALSE
  "CONFIGURACION"
  DoInputForm
  ITE
  ::
    ITE
    ::  BINT92  SetSysFlag  ;
    ::  BINT92  ClrSysFlag  ;
    ITE
    ::  BINT85  SetSysFlag  ;
    ::  BINT85  ClrSysFlag  ;
  ;
;
@
```

EJEMPLO 9: HP49 Menús y case.

```
::
  BINT0   "MENU1"  Str>Menu
  BINT22  "MENU2"  TRUE  Box/StdLabel  Grob>Menu
  BINT44
  GROB 0003A 8000051000000000002000003000002000002000002000007000000000
  Grob>Menu
  BINT66
  GROB 0003A 8000051000000000007000001000001000007000005000007000000000
  Grob>Menu
  BINT88  "HELP"
  TRUE  Box/StdLabel  Grob>Menu
  BINT110  "AQP"  Str>Menu
  WaitForKey
  DROP
  BINT1  #=casedrop  ::  BINT1  ;
  BINT2  #=casedrop  ::  BINT2  ;
  BINT3  #=casedrop  ::  BINT3  ;
  BINT4  #=casedrop  ::  BINT4  ;
  BINT5  #=casedrop  ::  BINT5  ;
  BINT6  #=case  ::  BINT6  ;
  DoBadKey
;
@
```