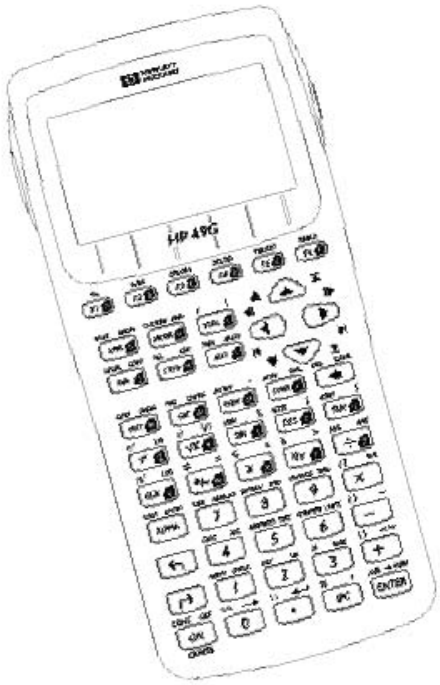
# PROGRAMACIÓN
# EN SYSTEM-RPL




by LUIS EDUARDO VALDIVIESO VIDAL

LOJA – ECUADOR

2 004

## 1.- PRÓLOGO.

El System-Rpl es un lenguaje que nos permite realizar cualquier programa que queramos, tales como los formularios del CALCULATOR MODES o TRANSFER, pero resulta muy peligroso al momento de correr un programa, al contrario del User, no detecta errores, causando la pérdida de archivos que han sido guardados en el puerto 0, 1 o Home.

Para programar en System-Rpl, existen algunos programas que nos permiten ensamblar el código system, tales como: el JAZZ, un excelente programa de Mika Heiskanen; la librería 256 y 257 que viene incorporada en la calculadora, El CQIF?, El Extable y El Emacs.

Yo en el tiempo que llevo programando en System-Rpl, siempre he utilizado el Emacs, aunque depende de cada persona acostumbrarse con cierto programa, es así que en el presente manual se utilizará el Emacs para ensamblar los programas, el mismo que requiere del Extable para su funcionamiento.

Este manual surge del estudio de diferentes documentos referentes al tema y de la experiencia adquirida durante todos estos años. Cuando no sabia programar en este lenguaje, no descanse hasta encontrar manuales que me permitiesen aprender, pero existía poca o nada bibliografía en español acerca del tema, es así que este manual se incorpora a la poca bibliografía existente.

Ahora quiero compartir todos mis conocimientos con mis compañeros de la Escuela de Ingeniería Civil de la Universidad Técnica Particular de Loja (U.T.P.L.).

No se olviden visitar mi sitio web, donde encontrarán información relacionada con la HP aplicada a Ingeniería Civil, además, pueden formar parte de nuestro equipo de programadores, único en el ECUADOR, desgraciadamente no existen otros sitios ECUATORIANOS, lo cual me entristece enormemente.

Como información adicional, les cuento que HP PROGRAMS CIVIL se encuentra en la red desde el 16 de Enero del 2 004, aunque estuvo planificado para el mes de Noviembre del año anterior, desafortunadamente por motivos ajenos a mi voluntad no ocurrió así.

Saludos desde ECUADOR a todos aquellos que visitan HP PROGRAMS CIVIL.

http://www.galeon.com/hpprograms
E-mail: luisv432latinmail.com

LUIS EDUARDO VALDIVIESO VIDAL
Loja, Marzo del 2 004

## 2.- PRINCIPIOS DEL RPL.

*(El siguiente extracto esta tomado textualmente de "RPL: Un Lenguaje de Control Matemático" por W.C. Wickes, publicado en "Entornos de Programación", Instituto para la Investigación de Forth Aplicado, Inc., 1988)*

En 1984, se inició un proyecto en la división de Hewlett-Packard en Corvallis para desarrollar el software de un nuevo sistema operativo para el desarrollo de una línea de calculadoras y dar soporte a una nueva generación de hardware y software. Anteriormente todas las calculadoras HP se implementaron enteramente en lenguaje ensamblador, un proceso que se iba haciendo cada vez m s pesado e ineficiente a medida que aumentaba la memoria de las calculadoras. Los objetivos para el nuevo sistema operativo fueron los siguientes:

- Proporcionar control de la ejecución y manejo de la memoria, incluyendo memoria conectable;
- Proporcionar un lenguaje de programación para un rápido desarrollo de prototipos y aplicaciones;
- Para soportar un variedad de calculadoras de negocio y técnicas;
- Para ejecutarse idénticamente en RAM y ROM;
- Para minimizar el uso de memoria, especialmente RAM;
- Para ser transportable a varias CPU's;
- Para ser extensible; y
- Para soportar operaciones de matemática simbólica.

Se tuvieron en cuenta varios lenguajes y sistemas operativos ya existentes pero ninguno cumplía con todos los objetivos del diseño. Por consiguiente se desarrolló un nuevo sistema, el cual mezcla la interpretación entrelazada de Forth con el enfoque funcional de Lisp.

El sistema operativo resultante, conocido de modo no oficial como RPL (de Reverse-Polish Lisp), hizo su primera aparición pública en junio de 1986 en la calculadora Business Consultant 18C. Mas tarde, RPL ha sido la base de las calculadoras HP-

17B, HP-19B, HP-27S, HP-28C y HP-28S y HP 48S y HP 48SX. La HP-17B, 18C y la 19B se diseñaron para aplicaciones de negocios; ellas y la calculadora científica HP-27S

ofrecen una lógica de cálculo "algebraica" y el sistema operativo subyacente es invisible para el usuario. Las familias HP 28/HP 48 de calculadoras

científicas usan una lógica RPN y muchas de la facilidades del sistema operativo están disponibles directamente como comandos de la calculadora.

# PARTE I

# TIPOS DE OBJETOS

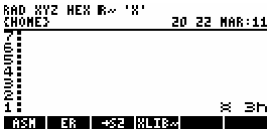
## 3.1.- TIPOS DE OBJETOS.

Antes de pasar a programar, es necesario conocer los tipos de objetos que posee la HP49G. Los tipos de objetos se los puede agrupar en tres clases:

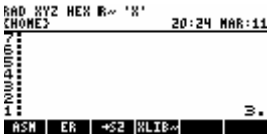
- Objetos de la Clase de Datos.
- Objetos de la Clase de Identificador.
- Objetos de la Clase de Procedimiento.

### 3.1.1 OBJETOS DE LA CLASE DE DATOS.

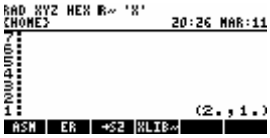
- OBJETO ENTERO BINARIO (BINTS).




- OBJETO REAL.



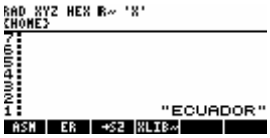

- OBJETO COMPLEJO.



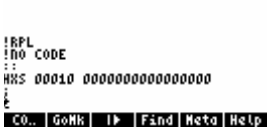

- OBJETO COMPLEJO EXTENDIDO.


- OBJETO FORMACIÓN.

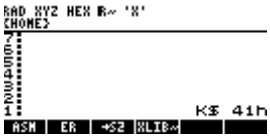

- OBJETO FORMACIÓN ENCADENADA.
- OBJETO CADENA DE CARACTERES.



- OBJETO CADENA HEX (CADENA HEXADECIMAL).

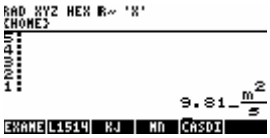



- OBJETO CARÁCTER




- OBJETO GRÁFICO (GROB).


OBJETO UNIDAD.

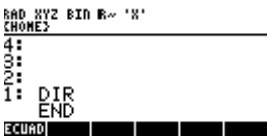



OBJETO LISTA.


- OBJETO SIMBÓLICO (EXPRESIÓN ALGEBRAICA).

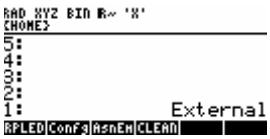

- OBJETO DATOS DE BIBLIOTECA.


OBJETO DIRECTORIO.



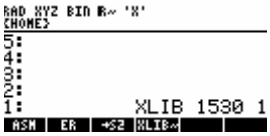
- OBJETO ETIQUETADO.


- OBJETO EXTERNO.




### 3.1.2 OBJETOS DE LA CLASE DE IDENTIFICADOR.

- OBJETO PUNTERO ROM (Nombre XLIB).




- OBJETO IDENTIFICADOR (NOMBRE GLOBAL).


- OBJETO IDENTIFICADOR TEMPORAL (NOMBRE LOCAL).


### 3.1.3 OBJETOS DE LA CLASE DE PROCEDIMIENTO.

- OBJETO CÓDIGO.


- OBJETO CÓDIGO PRIMITIVA.
- OBJETO PROGRAMA.


### 3.1.4 TERMINOLOGÍA Y ABREVIATURAS.

| OBJETO | ABREVIATURA |
|---|---|
| Cualquier objeto | ob |
| Objeto identificador | id |
| Objeto identificador temporal | lam |
| Objeto puntero ROM | romptr |
| Objeto entero binario | # |
| Objeto real | % |
| Objeto real extendido | %% |
| Objeto complejo | C% |
| Objeto complejo extendido | C%% |
| Objeto formación | arry |
| Objeto formación encadenada | lnkarry |
| Objeto cadena de caracteres | $ |
| Objeto cadena hex (cadena hexadecimal) | hxs |
| Objeto carácter | chr |
| Objeto externo | ext |
| Objeto código | code |
| Objeto código primitiva | primcode |
| Objeto secundario | : : |
| Objeto lista | list { } |
| Objeto simbólico | symb |
| Cualquier objeto compuesto | comp |
| Objeto directorio | rrp |
| Objeto etiquetado | tagged |
| TRUE/FALSE | flag |

## 4.- CREACIÓN DE OBJETOS.

Un número binario puede ser creado de diferentes formas:


Ó


Una vez compilado se tiene:


Un número entero se crea anteponiendo la palabra ZINT:


Un número real se representa anteponiendo el símbolo %:


Los números reales extendidos se representan anteponiendo el símbolo %%:


Los números complejos se representan anteponiendo el símbolo C%:


Un carácter se representa anteponiendo CHR_:


Las cadenas hexadecimales se crean usando la siguiente estructura:

| HXS <longitud> <hxscuerpo> |
|---|

La longitud esta en nibbles y hxscuerpo es el contenido de este.

## 4.1 CADENAS (STRINGS).

| | |
|---|---|
| $_R<< | (→"R<ángulo><ángulo>" ) |
| $_R<Z | (→ "R<ángulo>Z" ) |
| $_XYZ | (→ "XYZ") |
| $_<<>> | (→ "ABBB") |
| $_{} | (→ "{}") |
| $_[] | (→ "[]") |
| $_" | (→ """) |
| $_:: | (→ "::") |
| $_LRParens | (→ "()") |
| $_2DQ | (→ """""") |
| $_ECHO | (→ "ECHO") |
| $_EXIT | (→ "EXIT") |
| $_Undefined | (→ "Undefined") |
| $_RAD | (→ "RAD") |
| $_GRAD | (→ "GRAD") |

NEWLINE$ (→ "
")

SPACE$ (→ " ")

## 5.- CONVERSIÓN DE OBJETOS.

262F1 COERCE: Convierte un número real a binario ( % → #).


2F31F C%>#: Convierte un número complejo a binario ( C% → #).


05AO3 HXS># Convierte una cadena hexadecimal a binario ( hxs → # ).

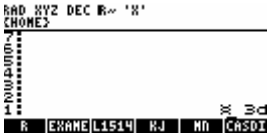
O5A51 CHR># Convierte un objeto a binario ( hxs → #).

059CC #>HXS Convierte un número binario a cadena hexadecimal ( # → hxs ).

2FFAP %>%%: Convierte un número real a real extendido ( % → %% ).


2FF9B %%>%: Convierte un número real extendido a real (%%→%).


262F6 UNCOERCE: Convierte un número binario a real (#→%).




%%P>R Convierte coordenadas polares (dadas en reales extendidas) a rectangulares reales ( %%radio %% ángulo → %%x %%y ).

%%R>P Convierte coordenadas rectangulares (dadas en reales extendidas) a polares reales ( %%x %%y → %%radio %% ángulo ).

COERCE2 Convierte dos números reales a binarios (%2 %1 → #2 #1).

COERCESWAP Convierte un real a binario y luego hace SWAP (ob %1 → #1 ob ).

UNCOERCE2 Convierte dos números binarios a reales (#2 #1 → %2 %1).

UNCOERCE%% Convierte un número binario a real extendido (# → %%).

%%H>HMS Horas decimales a hh.mmss ( %% --> %%' ).

## 6.- ARGUMENTOS.

En System-Rpl es muy importante chequear el tipo y número de argumentos que requiere un programa, cuando estos son ingresados por el usuario. No así con User-Rpl, pues aquí se hace el chequeo de los argumentos automáticamente.

Para chequear el número de argumentos de 0 a 5 presentes en la pila, use una de las siguientes ordenes:

| | |
|---|---|
| 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 |

Ahora para verificar el tipo de argumentos, se pueden utilizar los siguientes comandos:

- CK&DISPATCH0
- CK&DISPATCH1
- CK**N**&Dispatch

Los comandos CK&DISPATCH1 y CK&DISPATCH0 son usados para permitir que un programa realice acciones diferentes basados en los tipos de argumentos dados a él. Ellos se usan así:

```
::
--------
CK&DISPATCH1
          tipo1 acción1
         tipo2 acción2
         ---------------
          tipon acciónn
;
```

Los pares del tipo / acción son terminados por un ?SEMI (;). Si después de despachar, se requiere dar mas acciones para todos los tipos de argumentos, habrá que adjuntar todo el CK&DISPATCH1 en un bloque secundario (: : ; ).

El CK&DISPATCH0 trabaja así: verifica si la pila empareja las definiciones del tipo1. Si lo hace, se ejecuta la acción1 después de que el programa reasume la ejecución después de SEMI. Cada acción debe ser un solo objeto, si se quiere hacer mas de una acción, todos ellos deben ser incluidos en un secundario, es decir, entre : : y ;.

Si la definición del tipo no empareja la pila, entonces se verifica tipo2, y así sucesivamente. Si ningún tipo establecido fuese encontrado, ocurre un error, y se da un “Bad Argument Type” (“Tipo de Argumento Incorrecto”).

CK&DISPATCH1 procede así: Por cada tipo1, desde el tipo1 hasta el tipon, si el tipo1 coincide con la configuración de la pila, entonces se ejecuta la acción1, descartándose el resto de las palabras que contiene CK&DISPATCH1. Si no se encuentra ninguna coincidencia, se informa del error “Bad Argument Type”.

Incluso cuando su programa acepta una combinación de argumentos, este comando todavía es útil para verificar si los argumentos dados son del tipo.

La diferencia entre CK&DISPATCH1 y CK&DISPATCH0, es que este último, después de completar todos los pasos sin éxito, despoja todas las etiquetas de los argumentos, convierte enteros (ZINTS) a reales (REALES) y hace un segundo paso. Solo después del segundo paso sin una emparejada, devuelve "Bad Argument Type".
El entero binario tipo1 se codifica de la siguiente manera:

```
#nnnnn
|||||
||||+-- Tipo de argumento del nivel 1
|||+--- Tipo de argumento del nivel 2
||+---- Tipo de argumento del nivel 3
|+----- Tipo de argumento del nivel 4
+------ Tipo de argumento del nivel 5
```

Aquí cada "n" es un digito hexadecimal y representa el tipo de objeto, de acuerdo a la tabla V.1. Por ejemplo la definición de tipo #1982 representa un número real en el cuarto nivel, un objeto simbólico en el tercero, un objeto secundario en el segundo y un número complejo en el primer nivel de la pila. El tipo #AF indica una biblioteca de datos en el nivel 1.

**Tabla V.1:** TIPOS DE OBJETOS.

| Valor | Argumento | | # nombre | TYPE del Usuario |
|---|---|---|---|---|
| 0 | Cualquier Objeto | | any | |
| 1 | Número Real | real | | 0 |
| 2 | Número Complejo | | cmp | 1 |
| 3 | Cadena de Caracteres | | str | 2 |
| 4 | Formación | arry | | 3,4 |
| 5 | Lista | list | | 5 |
| 6 | Nombre Global | | idnt | 6 |
| 7 | Nombre Local | lam | | 7 |
| 8 | Secundario | seco | | 8 |
| 9 | Simbólico | symb | | 9 |
| A | Clase Simbólico | | sym | 6,7,9 |
| B | Cadena Hex | hxs | | 10 |
| C | Objeto Gráfico | grob | | 11 |
| D | Objeto Etiquetado | | TAGGED | 12 |
| E | Objeto Unidad | | unitob | 13 |
| 0F | Puntero ROM | | | 14 |
| 1F | Entero Binario | | | 20 |
| 2F | Directorio | | | 15 |
| 3F | Real Extendido | | | 21 |
| 4F | Complejo Extendido | | | 22 |
| 5F | Formación Encadenada | | | 23 |
| 6F | Carácter | | | 24 |

| | | | |
|---|---|---|---|
| 7F | Objeto Código | 25 | |
| 8F | Biblioteca | | 16 |
| 9F | Backup | 17 | |
| AF | Biblioteca de Datos | | 26 |
| BF | Objeto externo1 | 27 | |
| CF | Objeto externo2 | 28 | |
| DF | Objeto externo3 | 29 | |
| EF | Objeto externo4 | 30 | |

CK**N**&Dispatch es una combinación de CK**N** y CK&DISPATCH1, donde **N** es un número de 1 a 5, estas palabras combinan CK**N** con CK&DISPATCH1, porque ellos usan CK**N**, ellos solo deben ser usados en ordenes de biblioteca.

El comando TYPE se desmonta de la siguiente manera:

```
: :
CK1
  : : CK&DISPATCH0
        real                  %0
cmp                 %1
        str                   %2
        arry                  XEQTYPEARRY
        list                  %5
        id                    %6
        lam                   %7
        seco                  TYPESEC ( 8, 18, o 19 )
        symb                  %9
        hxs                   %10
        grob                  % 11
        TAGGED                % 12
        unitob                % 13
rompointer          % 14
        THIRTYONE ( # )       % 20
        rrp                   % 15
        # 3F ( %% )           % 21
        # 4F ( C%% )  % 22
        # 5F ( LNKARRY )      % 23
        # 6F ( CHR )          % 24
        # 7F ( CODE )         % 25
        library               % 16
backup              % 17
        # AF                  % 26 ( Biblioteca de Datos )
        any                   % 27 ( external )
  ;
SWAPDROP
;
```

**EJEMPLOS:**

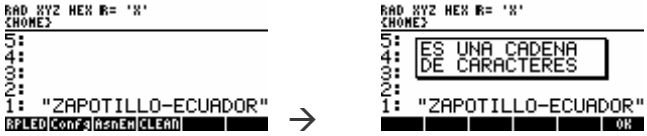
**Ej1.:** Verifica que haya un argumento en la pila.

```
: :
CK1NOLASTWD
;
@
```

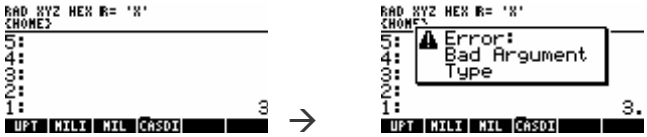
**Ej2.:**

```
: :
CK&Dispatch
#3
        : :
        “ES UNA CADENA DE CARACTERES”
        xMSGBOX
        ;
;
@
```

En este ejemplo si se pone una cadena devuelve el mensaje “ES UNA CADENA DE CARACTERES”.




Si se pone un número real, devuelve “Bad Argument Type”.




## 7.- CONDICIONANTES.

En System-Rpl los condicionantes son diferentes a los de User-Rpl, pues en el primero un “FALSO” se representa con un número 0, mientras que un “VERADERO” se representa con un número 1.

En System-Rpl, un “FALSO” se representa con un FALSO, y un “VERADERO” con un VERADERO.

Si se necesita convertir un VERDADERO o un FALSE a un número (1 o 0), puede emplearse el comando COERCEFLAG.

Los operadores boléanos como son NOT, AND, OR y XOR, son muy útiles cuando se requiere que se den mas de una condición.

## 7.1 BANDERAS Y TEST.

Estos test devuelven TRUE o FALSE, y sirvan para comparar dos objetos. Las siguientes palabras comprueban el tipo de objeto y la igualdad:

| | |
|---|---|
| = | Pregunta si dos objetos son iguales. |
| <= | Pregunta si el **ob2** es menor que el **ob1**. |
| < | Pregunta si el **ob2** es menor que el **ob1**. |
| > | Pregunta si el **ob2** es mayor que el **ob1**. |
| >= | Pregunta si el **ob2** es mayor o igual que el **ob1**. |
| <> | Pregunta si el **ob2** es diferente del **ob1**. |

Para poder comparar utilizar estos símbolos, debe especificar el tipo de objeto, por ejemplo, si se va a comparar dos números reales, se pondrá %=, %<, %<>, etc.

EQ ( ob1 ob2 → flag )
Si los objetos **ob1** y **ob2** son el mismo objeto, es decir, ocupan el mismo espacio en la memoria, entonces devuelve TRUE, si no FALSE.

EQUAL ( ob1 ob2 → flag )
Si el **ob1** es igual al **ob2**, devuelve TRUE, caso contrario FALSE, es similar a la palabra SAME del usuario.

2DUPEQ ( ob1 ob2 → ob1 ob2 flag )
Devuelve TRUE si **ob1** y **ob2** tienen la misma dirección física.

EQOR ( flag1 ob1 ob2 → flag2 )
Hace EQ y luego OR.

EQUALOR ( flag1 ob1 ob2 → flag2 )
Hace EQUAL y luego OR.

EQOVER ( ob1 ob2 ob3 → ob1 flag ob1 )
Hace EQ y luego OVER.

EQUALNOT ( ob1 ob2 → flag )
Devuelve FALSE si ob1 es igual a ob2.

Las siguientes palabras comprueban el tipo de un objeto. Las palabra de la forma TYPE...? tienen un diagrama de pila ( ob → flag ); las de la forma DTYPE...? o DUPTYPE...? duplican primero el objeto ( ob → ob flag ).

| Palabras Test | Tipo de Objeto |
|---|---|
| TYPEARRY? | Pregunta si es una formación. |
| DUPTYPEARRY? | Duplica y pregunta si es una formación. |
| TYPEBINT? | Pregunta si es un entero binario. |
| DUPTYPEBINT? | Duplica y pregunta si es un entero binario. |
| TYPECARRY? | Pregunta si es una formación compleja. |
| TYPECHAR? | Pregunta si es un carácter. |
| DUPTYPECHAR? | Duplica y pregunta si es un carácter. |
| TYPECMP? | Pregunta si es un número complejo. |
| DUPTYPECMP? | Duplica y pregunta si es un número complejo. |
| TYPECOL? | Pregunta si es un programa. |
| DUPTYPECOL? | Duplica y pregunta si es un programa. |
| TYPECSTR? | Pregunta si es una cadena. |
| DUPTYPECSTR? | Duplica y pregunta si es una cadena. |
| TYPEEXT? | Pregunta si es una unidad. |
| DUPTYPEEXT? | Duplica y pregunta si es una unidad. |
| TYPEGROB? | Pregunta si es un objeto gráfico. |
| DUPTYPEGROB? | Duplica y pregunta si es un objeto gráfico. |
| TYPEHSTR? | Pregunta si es una cadena hex. |
| DUPTYPEHSTR? | Duplica y pregunta si es una cadena hex. |
| TYPEIDNT? | Pregunta si es un identificador (nombre global). |
| DUPTYPEIDNT? | Duplica y pregunta si es un identificador (nombre global). |

| | |
|---|---|
| TYPELAM? | Pregunta si es un identificador local (nombre local). |
| DUPTYPELAM? | Duplica y pregunta si es un identificador local. |
| TYPELIST? | Pregunta si es una lista. |
| DUPTYPELIST? | Duplica y pregunta si es una lista. |
| TYPERARRY? | Pregunta si es una formación real. |
| TYPEREAL? | Pregunta si es un número real. |
| DUPTYPEREAL? | Duplica y pregunta si es un número real. |
| TYPEROMP? | Pregunta si es un puntero ROM (nombre XLIB). |
| DUPTYPEROMP? | |
| TYPERRP? | Pregunta si es un Directorio. |
| DUPTYPERRP? | Duplica y pregunta si es un directorio. |
| TYPESYMB? | Pregunta si es un objeto simbólico |
| DUPTYPESYMB? | Duplica y pregunta si es un objeto simbólico. |
| TYPETAGGED? | Pregunta si es un objeto etiquetado. |
| DUPTYPETAG? | Duplica y pregunta si es un objeto etiquetado. |

?SEMI ( flag --> )
Sale del programa en curso si la bandera es TRUE.

?SEMIDROP ( ob TRUE --> ) o ( FALSE --> )
Elimina ob si la bandera es TRUE; sale del programa en curso si la bandera es FALSE.

?SKIP ( flag --> )
Si la bandera es TRUE, se salta el siguiente objeto a continuación de ?SKIP.

NOT?SEMI ( flag --> )
Sale del programa en curso si la bandera es FALSE.

**Ejemplos:**

1. Aquí se va a comparar si el objeto del nivel 2 es igual al objeto del nivel.

: :

| | |
|---|---|
| CK2 | Requiere que hayan dos objetos en la pila. |
| %= | Pregunta si los números reales son iguales. |

;

@

2. Este programa compara si el objeto que se ingresa es un número real.

```
: :
CK1                 Requiere un argumento.
TYPERAL?            Pregunta si el objeto del nivel 1 es un número real.
;
@
```

### 7.2 If .. Then .. End.

Este es muy útil, cuando se requieren ejecutar dos programas, y cada uno depende de una condición.

La capacidad fundamental del If/Then/End, se proporciona por medio de las palabras RPITE y RPIT.

**RPITE** (flag ob1 ob2 → ?)
Si el flag es TRUE, se eliminan el flag, el **ob2** y se evalúa el **ob1**; y si el flag es FALSE, se eliminan el flag, el **ob1** y se evalúa el **ob2.**

**RPIT** (flag ob → ?)
Si el flag es TRUE, se elimina el flag y se evalúa el **ob**; y si el flag es FALSE, se eliminan flag y **ob**.

También tenemos las palabras IT e ITE:

La sintaxis del If .. Then .. End, es la siguiente:

```
: :
.
.
.
flag
IT
        : :
        Programa 1
        ;
        : :
        Programa 2
        ;

;
```

```
@
```

**También se puede emplear la palabra ITE, que en definitiva hacen lo mismo.**

**Ejemplos:**

1. Este ejemplo requiere dos números, si son iguales muestra el mensaje “SON IGUALES”, caso contrario “NO SON IGUALES”.

```
: :
CK2
%=



IT
        : :
        “SON IGUALES” xMSGBOX
        ;

        : :
        “NO SON IGUALES” xMSGBOX
        ;
;
@
```

2. Este programa requiere un objeto, y si el objeto es una matriz, devuelve TRUE, caso contrario FALSE.

```
: :
CK1
TYPEARRY?
ITE
        : :
        TRUE
        ;
        : :
        FALSE
        ;
;
@
```

### 7.3 Case

El CASE es una combinación de las palabras IT, SKIP, y COLA. La palabra básica es CASE, pero hay otras combinaciones de IT, test y otros comandos.

Si queremos construir una estructura que nos permita evaluar varios programas, podemos seguir el siguiente ejemplo:

```
: :
.
.
.
        : :
        DUP  #0=
        case
        : :
        Programa 1
        ;
        DUP  #1=
        case
        : :
        Programa 2
        ;
        DUP  #2=
        case
        : :


        Programa 3
        ;
        .
        .
        DUP  #n=
        Case
        : :
        Programa n
        ;
        .
        .
        .
        ;
.
.
```

```
.
;
@
```

Además podemos hacer el ejemplo anterior con la palabra OVER, así:

```
: :
.
.
.
        : :
        DUP
        BINT0 OVER#=case
        : :
        Programa 1
        ;
        DUP
        BINT1 OVER#=case
        : :
        Programa 2
        ;



        DUP
        BINT2 OVER#=case
        : :
        Programa 3
        ;
        .
        .
        DUP
        BINTn OVER#=case
        : :
        Programa n
        ;
        .
        .
        .
        ;
.
.
.
;
```

@

Palabras que hacen COLA o SKIP al siguiente objeto:

| | |
|---|---|
| #=casedrop | ( # # → )<br>( # #' → # )<br>Se debería llamar OVER#=casedrop |
| %1=case | ( % → ) |
| %0=case | ( % → flag ) |
| ANDNOTcase | ( flag1 flag2 → ) |
| ANDcase | ( flag1 flag2 → ) |
| case2drop | ( ob1 ob2 TRUE → )<br>( FALSE → ) |
| casedrop | ( ob TRUE → )<br>( FALSE → ) |
| DUP#0=case | ( # → # ) |
| DUP#0=csedrp | ( # → # ) # <> #0<br>( # → ) # = #0 |
| EQUALNOTcase | ( ob ob' → ) |
| EQUALcase | ( ob ob' → ) |
| EQUALcasedrp | ( ob ob' ob' → )<br>( ob ob' ob'' → ob ) |
| EQcase | ( ob1 ob2 → ) |
| NOTcase | ( flag → ) |
| NOTcasedrop | ( ob FALSE → )<br>( TRUE → ) |
| ORcase | ( flag1 flag2 → ) |

OVER#=case ( # #' → # )

Palabras case que o salen o continúan con el siguiente objeto:

caseDoBadKey ( flag → ) Sale via DoBadKey

caseDrpBadKey ( ob TRUE → ) Sale via DoBadKey
( FALSE → )

case2DROP ( ob1 ob2 TRUE → )
( FALSE → )

caseDROP ( ob TRUE → )
( FALSE → )

caseFALSE ( TRUE → FALSE )
( FALSE → )

caseTRUE ( TRUE → TRUE )
( FALSE → )

casedrpfls ( ob TRUE → FALSE )
( FALSE → )

case2drpfls ( ob1 ob2 TRUE → FALSE )
( FALSE → )

casedrptru ( ob TRUE → TRUE )
( FALSE → )

DUP#0=csDROP ( #0 → )
( # → # ) # <> 0.

NOTcaseTRUE ( FALSE → TRUE )
( TRUE → )

# PARTE II

# MANIPULACIÓN DE OBJETOS

# 8.- OPERACIONES ARITMÉTICAS.

## 8.1 NÚMEROS BINARIOS.

| | |
|---|---|
| #* | ( #2 #1 → #2*#1 ) |
| #+ | ( #2 #1 → #2+#1 ) |
| #+-1 | ( #2 #1 → #2+#1-1 ) |
| #- | ( #2 #1 → #2-#1 ) |
| #-#2/ | ( #2 #1 → (#2-#1)/2 ) |
| #-+1 | ( #2 #1 → (#2-#1)+1 ) |
| #/ | ( #2 #1 → #resto #cociente ) |
| #1+ | ( # → #+1 ) |
| #1+' | ( # → #+1 y ejecuta ' ) |
| #1+DUP | ( # → #+1 #+1 ) |
| #1- | ( # → #-1 ) |
| #10* | ( # → #*10 ) |
| #10+ | ( # → #+10 ) |
| #12+ | ( # → #+12 ) |
| #2* | ( # → #*2 ) |
| #2+ | ( # → #+2 ) |
| #2- | ( # → #-2 ) |
| #2/ | ( # → FLOOR(#/2) ) |
| #3+ | ( # → #+3 ) |
| #3- | ( # → #-3 ) |
| #4+ | ( # → #+4 ) |
| #4- | ( # → #-4 ) |
| #5+ | ( # → #+5 ) |
| #5- | ( # → #-5 ) |
| #6* | ( # → #*6 ) |
| #6+ | ( # → #+6 ) |
| #7+ | ( # → #+7 ) |
| #8* | ( # → #*8 ) |
| #8+ | ( # → #+8 ) |
| #9+ | ( # → #+9 ) |
| #MAX | ( #2 #1 → MAX(#2,#1) ) |
| #MIN | ( #2 #1 → MIN(#2,#1) ) |
| 2DUP#+ | ( #2 #1 → #2 #1 #1+#2 ) |
| DROP#1- | ( # ob → #-1 ) |
| DUP#1+ | ( # → # #+1 ) |
| DUP#1- | ( # → # #-1 ) |
| DUP3PICK#+ | ( #2 #1 → #2 #1 #1+#2 ) |
| OVER#+ | ( #2 #1 → #2 #1+#2 ) |
| OVER#- | ( #2 #1 → #2 #1-#2 ) |
| ROT#+ | ( #2 ob #1 → ob #1+#2 ) |

| | |
|---|---|
| ROT#+SWAP | ( #2 ob #1 → #1+#2 ob ) |
| ROT#- | ( #2 ob #1 → ob #1-#2 ) |
| ROT#1+ | ( # ob ob' → ob ob' #+1 ) |
| ROT+SWAP | ( #2 ob #1 → #1+#2 ob ) |
| SWAP#- | ( #2 #1 → #1-#2 ) |
| SWAP#1+ | ( # ob → ob #+1 ) |
| SWAP#1+SWAP | ( # ob → #+1 ob ) |
| SWAP#1- | ( # ob → ob #-1 ) |
| SWAP#1-SWAP | ( # ob → #-1 ob ) |
| SWAPOVER#- | ( #2 #1 → #1 #2-#1 ) |

## 8.2 NÚMEROS REALES.

| | |
|---|---|
| %* | ( %2 %1 → %2*%1 ) |
| %+ | ( %2 %1 → %2+%1 ) |
| %- | ( %2 %1 → %2-%1 ) |
| %/ | ( %2 %1 → %resto %cociente ) |
| %1+ | ( % → %+1 ) |
| %1- | ( % → %-1 ) |
| %10* | ( % → %*10 ) |
| %3+ | ( % → %+3 ) |
| %3- | ( % → %-3 ) |
| %4+ | ( % → %+4 ) |
| %4- | ( % → %-4 ) |
| %MAX | ( %2 %1 → MAX(%2,%1) ) |
| %MIN | ( %2 %1 → MIN(%2,%1) ) |

## 8.3 NÚMEROS REALES EXTENDIDOS.

| | |
|---|---|
| %%* | Multiplica dos reales extendidos ( %%1 %%2 → %%3 ). |
| %%*ROT | Multiplica dos reales extendidos y luego hace ROT ( ob1 ob2 %%1 %%2 → ob2 %%3 ob1). |
| %%*SWAP | Multiplica dos reales extendidos y luego hace SWAP ( ob %%1 %%2 → %%3 ob ). |
| %%*UNROT | Multiplica dos reales extendidos y luego hace UNROT ( ob1 ob2 %%1 %%2 → %%3 ob1 ob2 ). |
| %%+ | Suma dos reales extendidos ( %%1 %%2 → %%3 ) |

| | |
|---|---|
| %%- | Resta dos reales extendidos ( %%1 %%2 → %%3 ) |
| %%ABS | Valor absoluto de un real extendido ( %% → %%' ) |
| %%ACOSRAD | Arco-coseno de un real extendido ( %% → %%' ) |
| %%ANGLE | Angulo de un real extendido en el modo actual ( %%x %%y → %% ángulo ). |
| %%ANGLEDEG | Angulo de un real extendido usando grados sexagesimales ( %%x %%y → %% ángulo ). |
| %%ANGLERAD | Angulo en radianes ( %%x %%y → %% ángulo ). |
| %%ASINRAD | Arco-seno con radianes ( %% → %%' ). |
| %%CHS | Cambia el signo de un real extendido ( %% → %%' ). |
| %%COS | Coseno de un real extendido ( %% → %%' ). |
| %%COSDEG | Coseno con grados sexagesimales ( %% → %%' ). |
| %%COSH | Coseno hiperbólico ( %% → %%' ). |
| %%COSRAD | Coseno con radianes ( %% → %%' ). |
| %%EXP | Exponente ( %% → %%' ). |
| %%FLOOR | El mayor entero <=x ( %% --> %%' ). |
| %%INT | Devuelve la parte entera de un real extendido ( %% → %%' ). |
| %%LN | Logaritmo natural de un real extendido ( %% → %%' ). |
| %%LNP1 | ln(x+1) ( %% --> %%' ). |
| %%MAX | Devuelve el mayor numero de dos reales extendidos ( %%1 %%2 → %%3 ). |
| %%SIN | Seno de un real extendido ( %% → %%' ). |
| %%SINDEG | Seno con grados sexagesimales ( %% → %%' ). |
| %%SINH | Seno hiperbólico ( %% → %%' ). |
| %%SQRT | Raíz cuadrada ( %% → %%' ). |
| %%TANRAD | Tangente con radianes ( %% → %%' ). |

%%^ Exponencial ( %%1 %%2 → %%3 ).

## 9.- OPERACIONES DE LA PILA.

2DROP Borra dos objetos ( ob1 ob2 → ).

2DROP00 Borra dos objetos y pone #0 #0 ( ob1 ob2 → #0 #0 ).

2DROPFALSE Borra dos objetos y devuelve FALSE ( ob1 ob2 → FALSE ).

2DUP Duplica los objetos del nivel 1 y 2 ( ob1 ob2 → ob1 ob2 ob1 ob2).

2DUP5ROLL Duplica los objetos ( ob1 ob2 ob3 → ob2 ob3 ob2 ob3 ob1 ).

2DUPSWAP Duplica los objetos de los dos primeros niveles y luego intercambia los objetos de los niveles 1 y 2 ( ob1 ob2 → ob1 ob2 ob2 ob1 ).

2OVER ( ob1 ob2 ob3 ob4 → ob1 ob2 ob3 ob4 ob1 ob2 ).

2SWAP ( ob1 ob2 ob3 ob4 → ob3 ob4 ob1 ob2 ).

3DROP Borra los objetos de los tres primeros niveles ( ob1 ob2 ob3 → ).

3PICK ( ob1 ob2 ob3 → ob1 ob2 ob3 ob1 ).

3PICK3PICK ( ob1 ob2 ob3 → ob1 ob2 ob3 ob1 ob2 ).

3PICKOVER ( ob1 ob2 ob3 → ob1 ob2 ob3 ob1 ob3 ).

3PICKSWAP ( ob1 ob2 ob3 → ob1 ob2 ob1 ob3 ).

3UNROLL ( ob1 ob2 ob3 → ob3 ob1 ob2 ).

4DROP ( ob1 ob2 ob3 ob4 → ).

4PICK ( ob1 ob2 ob3 ob4 → ob1 ... ob4 ob1 ).

4PICKOVER ( ob1 ob2 ob3 ob4 → ob1 ob2 ob3 ob4 ob1 ob4 ).

4PICKSWAP ( ob1 ob2 ob3 ob4 → ob1 ob2 ob3 ob1 ob4 ).

4ROLL ( ob1 ob2 ob3 ob4 → ob2 ob3 ob4 ob1 ).

4UNROLL ( ob1 ob2 ob3 ob4 → ob4 ob1 ob2 ob3 ).

4UNROLL3DROP ( ob1 ob2 ob3 ob4 → ob4 ).

4UNROLLDUP ( ob1 ob2 ob3 ob4 → ob4 ob1 ob2 ob3 ob3 ).

4UNROLLROT ( ob1 ob2 ob3 ob4 → ob4 ob3 ob2 ob1 ).

5DROP ( ob1 ... ob5 → ).

5PICK ( ob1 ... ob5 → ob1 ... ob5 ob1 ).

5ROLL ( ob1 ... ob5 → ob2 ... ob5 ob1 ).

5ROLLDROP ( ob1 ... ob5 → ob2 ... ob5 ).

5UNROLL ( ob1 ... ob5 --> ob5 ob1 ... ob4 ).

6DROP ( ob1 ... ob6 → ).

6PICK ( ob1 ... ob6 → ob1 ... ob6 ob1 ).

6ROLL ( ob1 ... ob6 → ob2 ... ob6 ob1 ).

7DROP ( ob1 ... ob7 → ).

7PICK ( ob1 ... ob7 → ob1 ... ob7 ob1 ).

7ROLL ( ob1 ... ob7 → ob2 ... ob7 ob1 ).

8PICK ( ob1 ... ob8 → ob1 ... ob8 ob1 ).

8ROLL ( ob1 ... ob8 → ob2 ... ob8 ob1 ).

8UNROLL ( ob1 ... ob8 → ob8 ob1 ... ob7 ).

DEPTH ( ob1 ... obn ... → #n ).

DROP ( ob → ).

DROPDUP ( ob1 ob2 → ob1 ob1 ).

DROPFALSE ( ob → FALSE ).

DROPNDROP ( ... # ob ) elimina ob, luego DROP # objetos

DROPONE ( ob → #1 ).

DROPOVER ( ob1 ob2 ob3 → ob1 ob2 ob1 ).

DROPRDROP ( ob → ) DROP ob y baja 1 nivel de la pila de retornos.

DROPROT ( ob1 ob2 ob3 ob4 → ob2 ob3 ob1 ).

DROPSWAP ( ob1 ob2 ob3 → ob2 ob1 ).

DROPSWAPDROP ( ob1 ob2 ob3 → ob2 ).

DROPTRUE ( ob → TRUE ).

DROPZERO ( ob → #0 ).

DUP ( ob → ob ob ).

DUP#1+PICK ( ... #n → ... #n obn ).

DUP3PICK ( ob1 ob2 → ob1 ob2 ob2 ob1 ).

DUP4UNROLL ( ob1 ob2 ob3 → ob3 ob1 ob2 ob3 ).

DUPDUP ( ob → ob ob ob ).

DUPONE ( ob → ob ob #1 ).

DUPPICK ( ... #n → ... #n obn-1 ).

DUPROLL ( ... #n → ... #n obn-1 ).

DUPROT ( ob1 ob2 → ob2 ob2 ob1 ).

DUPTWO ( ob → ob ob #2 ).

DUPUNROT ( ob1 ob2 → ob2 ob1 ob2 ).

DUPZERO ( ob → ob ob #2 ).

N+1DROP ( ob ob1 ... obn #n → ).

NDROP ( ob1 ... obn #n → ).

NDUP ( ob1 ... obn #n → ob1 ... obn ob1 ... obn ).

NDUPN ( ob #n → ob ... ob ).

ONEFALSE ( → #1 FALSE ).

ONESWAP ( ob → #1 ob ).

OVER ( ob1 ob2 → ob1 ob2 ob1 ).

OVER5PICK ( v w x y z → v w x y z y v ).

OVERDUP ( ob1 ob2 → ob1 ob2 ob1 ob1 ).

OVERSWAP ( ob1 ob2 → ob2 ob1 ob1 ).

OVERUNROT ( ob1 ob2 → ob1 ob1 ob2 ).

PICK ( obn ... #n → ... obn ).

ROLL ( obn ... #n → ... obn ).

ROLLDROP ( obn ... #n → ... ).

ROLLSWAP ( obn ... ob #n --> ... obn ob ).

ROT ( ob1 ob2 ob3 → ob2 ob3 ob1 ).

ROT2DROP ( ob1 ob2 ob3 → ob2 ).

ROT2DUP ( ob1 ob2 ob3 → ob2 ob3 ob1 ob3 ob1 ).

ROTDROP ( ob1 ob2 ob3 → ob2 ob3 ).

ROTDROPSWAP ( ob1 ob2 ob3 → ob3 ob2 ).

ROTDUP ( ob1 ob2 ob3 → ob2 ob3 ob1 ob1 ).

ROTOVER ( ob1 ob2 ob3 → ob2 ob3 ob1 ob3 ).

ROTROT2DROP ( ob1 ob2 ob3 → ob3 ).

ROTSWAP ( ob1 ob2 ob3 → ob2 ob1 ob3 ).

SWAP ( ob1 ob2 → ob2 ob1 ).

SWAP2DUP ( ob1 ob2 → ob2 ob1 ob2 ob1 ).

SWAP3PICK ( ob1 ob2 ob3 → ob1 ob3 ob2 ob1 ).

SWAP4PICK ( ob1 ob2 ob3 ob4 → ob1 ob2 ob4 ob3 ob4 ).

SWAPDROP ( ob1 ob2 → ob2 ).

SWAPDROPDUP ( ob1 ob2 → ob2 ob2 ).

SWAPDROPSWAP ( ob1 ob2 ob3 → ob3 ob1 ).

SWAPDROPTRUE ( ob1 ob2 → ob2 TRUE ).

SWAPDUP ( ob1 ob2 → ob2 ob1 ob1 ).

SWAPONE ( ob1 ob2 → ob2 ob1 #1 ).

SWAPOVER ( ob1 ob2 → ob2 ob1 ob2 ).

SWAPROT ( ob1 ob2 ob3 → ob3 ob2 ob1 ).

SWAPTRUE ( ob1 ob2 → ob2 ob1 TRUE ).

UNROLL ( ... ob #n → ob ... ).

UNROT ( ob1 ob2 ob3 → ob3 ob1 ob2 ).

UNROT2DROP ( ob1 ob2 ob3 → ob3 ).

UNROTDROP ( ob1 ob2 ob3 → ob3 ob1 ).

UNROTDUP ( ob1 ob2 ob3 → ob3 ob1 ob2 ob2 ).

UNROTOVER ( ob1 ob2 ob3 → ob3 ob1 ob2 ob1 ).

UNROTSWAP ( ob1 ob2 ob3 → ob3 ob2 ob1 ).

ZEROOVER ( ob → ob #0 ob ).

reversym ( ob1 ... obn #n → obn ... ob1 #n ).

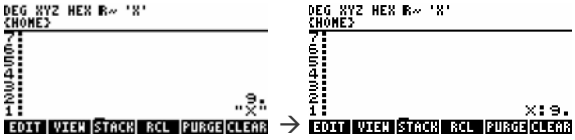
## 10.- MANIPULACIÓN DE OBJETOS.

### 10.1 OBJETOS ETIQUETADOS.

Use la estructura TAG <tag> <object> para crear objetos etiquetados. Las siguientes palabras están disponibles para manipular objetos etiquetados.

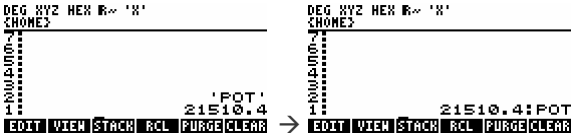
05E81 >TAG (ob $ → tagged)

Etiqueta el ob2 con el la cadena (string) del nivel 1.




2F223 %>TAG ( ob % → tagged )

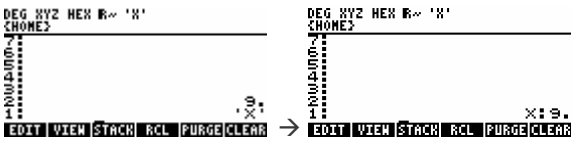
Etiqueta el objeto del nivel 2 con el número real del nivel 1.




05F2E ID>TAG ( ob id/lam → tagged )

( ob ... {$ ...} → tagged ...)

Etiqueta el objeto del nivel 2 con la variable global/temporal del nivel 1.




37ABE STRIPTAGS ( tagged → ob )

Elimina todas las etiquetas.

37AEB STRIPTAGSl2 ( tagged ob' → ob ob' )
Quita las etiquetas del objeto etiquetado del nivel 2.

37BO4 TAGOBS ( ob $ → tagged )
( ob1... obn {$1 ... $n} → tagged_1 ... tagged_n)
Etiqueta un objeto, o varios objetos si hay una lista de etiquetas en el nivel 1.

2F266 USER$>TAG ( ob $ → tagged )
Etiqueta ob con $ (válido hasta 255 caracteres).

## 10.2 OBJETOS UNIDADES.

U>NCQ ( unidad → n%% cf%% qhxs )
Devuelve el número, el factor de conversión y la cadena de cantidad hex (qhxs).

UM=? ( unidad1 unidad2 → %flag )
Devuelve %1 si dos obs unidades son iguales.

UM#? ( unidad1 unidad2 → %flag )
Devuelve 1% si unidad1 <> unidad2.

UM<? ( unidad1 unidad2 → %flag )
Devuelve %1 si unidad1 < unidad2.

UM>? ( unidad1 unidad2 → %flag )
Devuelve %1 si unidad1 > unidad2.

UM<=? ( unidad1 unidad2 → %flag )
Devuelve %1 si unidad1 <= unidad2.

UM>=? ( unidad1 unidad2 → %flag )
Devuelve %1 si unidad1 >= unidad2.

UM>U ( % unidad → unidad' )
Reemplaza la parte numérica de un objeto unidad.

UM% ( unidad %porcentaje → unidad' )
Devuelve un porcentaje de un objeto unidad.

UM%CH ( unidad1 unidad2 → % )
Devuelve la diferencia porcentual.

UM%T ( unidad1 unidad2 → % )

Devuelve la fracción porcentual.

UM+ ( unidad1 unidad2 → unidad3 )
Suma.

UM- ( unidad1 unidad2 → unidad3 )
Resta.

UM* ( unidad1 unidad2 → unidad3 )
Multiplicación.

UM/ ( unidad1 unidad2 → unidad3 )
División.

UM^ ( unidad1 unidad2 → unidad3 )
Potencia.

UM1/ ( unidad → unidad' )
Inverso.

UMABS ( unidad → unidad' )
Valor absoluto.

UMCHS ( unidad → unidad' )
Cambio de signo.

UMCONV ( unidad1 unidad2 → unidad1' )
Convierte unidad1 a las unidades de unidad2.

UMCOS ( unidad → unidad' )
Coseno.

UMMAX ( unidad1 unidad2 → unidad? )
Devuelve la mayor de unidad1 y unidad2.

UMMIM ( unidad1 unidad2 → unidad? )
Devuelve la menor de unidad1 y unidad2.

UMSI ( unidad → unidad' )
Convierte a las unidades básicas SI.

UMSIN ( unidad → unidad' )
Seno.

UMSQ ( unidad → unidad' )
Cuadrado.

UMSQRT ( unidad → unidad' )
Raíz cuadrada.

UMTAN ( unidad → unidad' )
Tangente.

UMU> ( unidad → % unidad' )
Devuelve la parte numérica y la parte normalizada de unidad de un objeto de unidad.

UMXROOT ( unidad1 unidad2 → unidad3 )
unidad1^1/unidad2.

UNIT>$ ( unidad → $ )
Descompila un objeto unidad a una cadena de texto.

## 11.- VARIABLES Y DIRECTORIOS.

Existen dos tipos de variables las variables temporales o locales y las variables globales.

**11.1 VARIABLES LOCALES (LAMs).-** Son aquellas que solo pueden ser utilizadas en un mismo programa. Ocupan menos espacio.

074D0 BIND Crea un nuevo entorno de variables temporales.

07497 ABND Abandona el entorno temporal.

O75A5 GETLAM Devuelve el objeto (contenido) de la enésima variable( #n → ob ).

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

STO Almacena un objeto en la variable global/temporal con nombre ( ob id → ).

2B3AB NULLLAM Pone NULLLAM en la pila ( → NULLLAM ).

35DEE 1ABNDSWAP Hace :: 1GETLAM ABND SWAP ; ( ob → lamob ob ).

364FF 1GETABND Hace :: 1GETLAM ABND ; ( → lamob ).

34616 1GETLAM ( → ob ).

346E8 22GETLAM Devuelve el contenido del enésimo lam.

35F42 1GETSWAP Hace :: 1GETLAM SWAP ; ( ob → lamob ob )

36518 1LAMBIND Hace :: 1NULLLAM{} BIND ; ( ob → ).

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

34611 1PUTLAM ( ob → ).

346E3 22PUTLAM Almacena un objeto en el enésimo lam.

3632E 2GETEVAL Llama y evalúa el ob en el segundo lam ( → ? ).

07943 @LAM Llama lam por nombre, devuelve ob y TRUE si id existe; si no, FALSE
( id → ob TRUE )
( id → FALSE ).

34D00 CACHE 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 ( obn ... ob1 n lam→).

34EBE DUMP DUMP es esencialmente el inverso de CACHE, PERO SOLO funciona cuando el nombre cacheado es NULLLAM y SIEMPRE hace una recolección de basura ( NULLLAM → ob1..obn n).

36513 DUP1LAMBIND Hace DUP y luego 1LAMBIND ( ob → ob ).

34797 DUP4PUTLAM Hace DUP y luego 4PUTLAM ( ob → ob ).

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

```
: :
%1                          Pone el numero 1 en el nivel 2
%2                          Pone el numero 2 en el nivel 1
{ LAM N1
 LAM N2
}
```

```
BIND
LAM N1                  Devuelve 1 desde N1
LAM N2                  Devuelve 2 desde N2
%*                          Multiplica 1 y 2
ABND                        Abandona el entorno temporal
;
@
```

También podemos utilizar nombres nulos, ahorrando así memoria, el ejemplo anterior seria así:

```
: :
%1                          Pone el numero 1 en el nivel 2
%2                          Pone el numero 2 en el nivel 1
{ NULLLAM
  NULLLAM
}
```

```
BIND
1GETLAM                     Devuelve 1 desde la primera variable
2GETLAM                     Devuelve 2 desde la segunda variable
%*                          Multiplica 1 y 2
ABND                        Abandona el entorno temporal
;
@
```

Si se necesita utilizar un gran número de variables temporales, podemos hacerlo de la siguiente manera:

```
: :
CK5                         Verifica que hayan 5 argumentos en la pila
NULLLAM
FIVE
NDUPN
{}N
BIND
:
:
:
ABND
;
@
```

11.2 **VARIABLES GLOBALES (IDs).-** Son aquellas que pueden ser utilizadas de un programa a otro. Ocupan mas espacio.

```
: :
%1                          Pone el numero 1 en el nivel 2
%2                          Pone el numero 2 en el nivel 1
ID N1
STO
ID N2
STO
ID N1                       Devuelve 2 desde N1
ID N2                       Devuelve 1 desde N2
%*                          Multiplica 1 y 2
;
@
```

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

CREATE (ob id → )

Crea una variable con el nombre **id** y contenido **ob.** Un error ocurriría si **ob** es o contiene el directorio actual (“Directorio Recursivo”). Esta palabra no chequea si hay ya una variable con el mismo **id,** aún cuando hay no se crea otro.

STO (ob id → )

(ob lam → )

En el caso **lam**, el identificador **lam** se vuelve a unir al Nuevo objeto. Un error ocurre si el **lam** no esta unido a nada. En el caso id, STO intenta emparejar **id** con la parte nombre parte nombre de una variable global.

@ ( id → ob TRUE )

( id → FALSE )

( lam → ob TRUE )

( lam → FALSE )

Intenta retornar el contenido guardado en la variable o identificador temporal. Retorna el objeto guardado y TRUE si lo consigue, o FALSE si ninguna variable o **lam** se encontraron con ese

nombre. En el caso de variables, empieza buscando en el directorio actual y a través de directorios del home.

CONTEXT! ( rrp → )
Almacena como directorio actual un puntero a un directorio enraizado.

CONTEXT@ (→ rrp )
Llama a la pila el directorio contexto actual.

CREATEDIR ( id → )
Crea un objeto directorio en el directorio actual.

DOVARS (→ { id1 id2 ... } )
Devuelve la lista de los nombres de variables del directorio actual.

HOMEDIR (→ )
Hace de HOME el directorio actual.

PATHDIR (→ { HOME dir dir ... } )
Devuelve el camino actual.

UPDIR (→ )
Sube un directiorio.

XEQORDER ( { id1 id2 ... } → )
Ordena el directorio actual.

XEQPGDIR ( id --> )
Elimina un directorio mientras respeta las convenciones de referencia/recolección de basura.

### El Directorio Oculto (Hidden Directory).

Este directorio se encuentra al principio del directorio “Home” y no tiene nombre. Este directorio contiene las definiciones de teclas del usuario y la información de las alarmas.

La siguientes palabras proporcionan capacidades de almacenamiento, llamada y eliminación de variables en el directorio oculto.

SetHiddenRes (→ )
Establece el directorio oculto como el directorio actual y al final de los directorios.

WithHidden (→ ? )
Ejecuta el siguiente objeto en el directorio oculto.

PuHiddenVar ( id → )
Elimina la variable oculta de nombre id.

RclHiddenVar ( id → ob TRUE )
( id → FALSE )
Llama a una variable oculta.

StoHiddenVar ( ob id --> )
Almacena **ob** en una variable oculta.

## 12.- FUNCIONES DEL SISTEMA (FLAGS).

2614D SetSysFlag: Activa el flag indicado del sistema (# → ).
26044 ClrSysFlag: Borra el falg indicado del sistema (# → ).
26170 TestSysFlag: Retorna TRUE si el flag esta activado (#→ TRUE).
26152 SetUserFlag: Activa el flag indicado del usuario (# → ).
26049 ClrUserFlag: Borra el flag indicado del usuario (# → ).
26175 TestUserFlag: Retorna TRUE si el flag del usuario esta activado (# → TRUE).
25F23 SaveSysFlags: Guarda los flags del sistema en el virtual stack ( → ).
25F22 RestoreSysFlags: Restaura los flags del sistema (→ ).
2EFA5 DOHEX: Pone la base en modo hexadecimal.
2EFA9 DODEC: Pone la base en modo decimal.
2EFA6 DOBIN: Pone la base en modo binario.
2EFA7 DOOCT: Pone la base en modo octal.
26053 DOSTD: Pone la pantalla en modo estándar.
26059 DOFIX: Pone la pantalla en modo FIX, que va desde 0 a 11 dígitos decimales. (#→).
26058 DOSCI: Pone la pantalla en modo científico, y va desde 0 a 11 dígitos (#→).
2604E DOENG: Pone la pantalla en modo de ingeniería, va desde 0 a 11 dígitos (# → ).

Ej.:

- Este programa activa el flag 40 que corresponde al reloj. Si el reloj esta activado permanece activado.

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

# 13.- BUCLES.

Existen dos tipos de bucles, los bucles indefinidos y los bucles definidos.

## 13.1 BUCLES INDEFINIDOS.

Los bucles indefinidos se pueden construir a partir de las siguientes palabras:

**BEGIN**
Copia el puntero del interprete en la pila de retorno.

**UNTIL**
Si el flag es TRUE, elimina el puntero de la cima de la pila de retornos, caso contrario copia ese puntero al puntero del interprete.

**WHILE**
Si el flag es TRUE entonces no realiza nada. Caso contrario elimina el primer puntero de la línea de retornos y hace que el puntero del interprete se salte los dos siguientes objetos.

**REPEAT**
Copia el primer puntero de la pila de retornos al puntero del interprete.

**AGAIN**
Copia el primer puntero de la pila de retornos al puntero del interprete.

El primer bucle indefinido es el bucle WHILE y esta creado así:

BEGIN
< cláusula de prueba o test >
WHILE
< objeto del bucle >
REPEAT

Este tipo de bucle ejecuta la <cláusula de test>, y si el test es TRUE, el <objeto del bucle> es ejecutado y la vuelta empieza de nuevo Si el test retorna FALSE, entonces sale y sigue justo pasado el REPEAT. Si el primer test volviera FALSE, el bucle WHILE nunca se ejecutaría.

Este bucle requiere que el <objeto del bucle> sea un solo objeto.

El segundo tipo de bucle indefinido es el UNTIL, y se crea de la siguiente manera:

BEGIN
< cláusula del bucle >
UNTIL

Este bucle siempre se ejecuta por lo menos una vez. La palabra UNTIL espera un flag. Si el flag es FALSE, el <cláusula del bucle> se ejecuta otra vez. Si el flag es TRUE, la ejecución continua pasado UNTIL.

Un tercer bucle indefinido es BEGIN ... UNTIL

BEGIN
< cláusula del bucle >
AGAIN

Este bucle no tiene un test. Para terminarlo se requiere que suceda una condición de error o manipular directamente la pila de retorno. Esto es útil si el código del bucle contiene varias situaciones diferentes a que las decisiones sobre repetir o terminar el bucle tienen que ser hechas.

## 13.2 BUCLES DEFINIDOS.

Los bucles definidos son creados con las palabras **DO** y **Loop** u otras palabras semejantes. DO toma dos números binarios de la pila, representando el índice y el valor de parada. El índice mas interno (superior) es llamado con INDEX@ y el índice en el segundo entorno se llama con JINDE@ . El valor de parada mas interno (superior) esta disponible con ISTOP@.

Los complementos de DO son LOOP y +LOOP. La palabra LOOP incrementa el valor del DoLoop mas interno, y si el valor nuevo es mayor o igual al valor de parada, LOOP elimina el puntero de la cima de la pila de retornos y elimina el entorno DoLoop mas interno. Caso contrario LOOP copia el puntero de la cima de la pila de retornos al puntero del interprete.

+LOOP trabaja similarmente a LOOP, solo que este incrementa el contador del bucle en una cantidad (número entero binario) tomado de la pila.

La forma estándar de un DoLoop es:

parada inicio DO < cláusula del bucle > LOOP

que ejecuta <cláusula del bucle> para cada valor del índice desde inicio hasta parada –1.

Note que aquí el valor de parada es mayor en #1, y que el valor de parada va antes que el valor de entrada.

073DB #1+_ONE_DO * ( #fin --> )
Equivalente a #1+ ONE DO; usada a menudo para ejecutar un bucle #fin veces.

073F7 DO ( #fin #inicio --> )
Empieza el bucle DO.

346AF DROPLOOP * ( ob --> )
Hace un DROP y luego LOOP.

364C8 DUP#0_DO * ( # --> # )
Empieza el bucle # ...#0 DO.

3645A DUPINDEX@ ( ob --> ob ob #índice )
Hace DUP, luego devuelve el valor del índice del entorno DoLoop mas interno.

3709B ExitAtLOOP ( --> )
Almacena cero en el valor de parada del entorno DoLoop más interno.

07221 INDEX@ ( --> #índice )
Devuelve el índice del entorno DoLoop mas interno.

367D9 INDEX@#- ( # --> #' )
Resta a # el valor del índice del entorno DoLoop mas interno.

07270 INDEXSTO ( # --> )
Almacena # como el índice del entorno DoLoop mas interno.

07249 ISTOP@ ( --> #stop )
Devuelve el valor de parada del entorno DoLoop mas interno.

07295 ISTOPSTO ( # --> )
Almacena el nuevo valor de parada en el el entorno DoLoop mas interno.

07258 JINDEX@ ( --> #índice )
Devuelve el índice del segundo entorno DoLoop.

07334 LOOP ( --> )
Termina una estructura de bucle.

NOT_UNTIL * ( flag --> )
Termina una estructura de bucle.

073CE ONE_DO * ( #fin --> )
Comienza un Bucle #1...#fin DO.

36482 OVERINDEX@ ( ob1 ob2 --> ob1 ob2 ob1 #índice )
Hace OVER y luego devuelve el valor del índice del entorno DoLoop mas interno.

3646E SWAPINDEX@ ( ob1 ob2 --> ob2 ob1 #índice )
Hace SWAP y luego devuelve el valor del índice del entorno DoLoop mas interno.

36496 SWAPLOOP * ( ob1 ob2 --> ob2 ob1 )
Hace SWAP y luego LOOP

ZEROISTOPSTO ( --> )

Almacena cero como el valor de parada del entorno DoLoop mas interno.

073CE ZERO_DO * ( #fin --> )

Comienza el bucle DO desde #0 hasta #fin.

364E1 toLEN_DO ( {list} --> {list} )

Comienza el bucle DO con inicio = #1 y con el valor de parada = #número-de-elementos-de-la-lista+1

**Ejemplos:**

El siguiente programa va a poner en la pila los números del 1 al 10.

: :
CK0
BINT11
ONE_DO
INDEX@
UNCOERCE
LOOP
;
@

# PARTE III

# ENTRADA DE DATOS

## INTRODUCCIÓN.

El System-Rpl, permite ingresar datos de una forma muy cómoda, es así que existen varias formas de ingresar datos, desde las mas sencillas hasta las mas complejas.

Entre las sencillas tenemos ingresando directamente en la pila, el InputLine; hasta las mas complejas como son los formularios de entrada (DoInputForm), el Bucle Externo Parametrizado (PartOuterLoop), el HP48 Browser, el Filer Manager, etc.

Para ingresar datos directamente en la pila se requiere necesariamente indicar al principio del programa la cantidad y el tipo de objeto.

Por ejemplo, el siguiente programa requiere que ingresemos dos números reales para su funcionamiento.

```
: :
CK2&Dispatch
#11
.
.
.
;
@
```

El siguiente ejemplo requiere que en el nivel tres haya una lista, y en el nivel dos y uno haya un numero real.

```
: :
CK3&Dispatch
#511
.
.
.
;
@
```

## 14.-EL InputLine

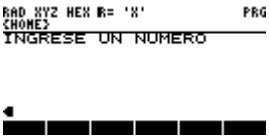
El comando InputLine del sistema equivale al comando del usuario INPUT . Su uso es similar, y hace lo siguiente:

- Muestra un mensaje en la parte superior de la pantalla.
- Empieza los modos de entrada del teclado.
- Empieza la línea de edición.
- Acepta la entrada, si se presiono ENTER.
- Analiza, evalúa o devuelve la entrada del usuario.
- Devuelve TRUE si es terminado por ENTER, o FALSE si fue abortado por ON/CANCEL.

Requiere los siguientes argumentos para su funcionamiento:

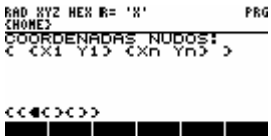
1. Un mensaje en cadena (“mensaje”).
2. Valor inicial que aparecerá, en cadena (“Valor inicial”). Si no queremos ningún valor, podemos escribir NULL$ o “”.
3. Posición del cursor, en enteros binarios. El #0 indica el cursor al final de la línea de edición.
4. Modo de escritura-sobreescritura, de la siguiente manera:
   - #0 Escritura normal.
   - #1 Modo de escritura normal.
   - #2 Modo de sobreescritura.
5. Modo de entrada inicial.
   - #0 Modo de entrada actual mas el modo de entrada del programa.
   - #1 Solo modo de entrada del programa.
   - #2 Programa y modos de entrada algebraico.
6. Modo de alfa inicial.
   - #0 Modo actual.
   - #1 Alfa habilitado.
   - #2 Alfa deshabilitado.
7. El menú inicial. Se refiere al menú que queramos que aparezca, para lo cual debe estar en cadena, si no queremos ningún menú, se coloca NULL{}.
8. El menú inicial del numero de fila, normalmente se pone BINT1, para mostrar el primero de la pagina.
9. Un flag.
   - TRUE abortara la entrada cuando se presione CANCEL.
   - FALSE borrara la línea de edición cuando se presione CANCEL.
10. Proceso de edición de la línea.
    - #0 Devuelve la línea de edición en una cadena.
    - #1 Devuelve la línea de edición como una cadena y un objeto analizado.
    - #2 Analiza y evalúa la línea de edición.
11. InputLine.

Ej. 1:



```
: :
CK0NOLASTWD                 No requiere argumentos
“INGRESE UN NUMERO”         Mensaje a mostrar
NULL$                       Valor inicial
BINT1                       Posición del cursor
BINT1                       Modo de escritura normal
BINT1                       Modo de entrada del programa
BINT0                       Modo actual
NULL{}                      No queremos ningún menú
BINT1                       Primera fila del menú
FALSE                       Borrara la línea de edición al presionar CANCEL
BINT2                       Devuelve la línea de edición analizada
InputLine                   Ejecuta el InputLine
NOT?SEMI                    Sale del programa si se presiona CANCEL
;
@
```

Ej. 2:




```
: :
CK0NOLASTWD
“COORDENADAS NUDOS:
{ { X1 Y1} {Xn Yn} }”
“{{}{}{}}”
BINT3
BINT0
BINT1
BINT2
NULL{}
BINT1
FALSE
BINT1
InputLine
NOT?SEMI
;
@
```

*Note la posición del cursor en ambos casos.

## 15.- FORMULARIOS DE ENTRADA (DoInputForm)

Como dije en la Introducción, con este comando del sistema podemos crear formularios de entrada de datos, tales como el CALCULATORS MODES o el TRANSFER.

En User también se pueden crear formularios de entrada con el comando INFORM, pero estos son limitados, caso contrario con System, en donde podemos crear formularios con campos CHOOSE, CHK, y en System son mas rápidos de cargarse.

El DoInputForm requiere varios argumentos, y se han dividido en tres partes:

1. **DEFINICIÓN DE ETIQUETAS:** Cada etiqueta requiere tres argumentos.
   "Nombre"
   Coordenada #X
   Coordenada #Y

2. **DEFINICIÓN DE CAMPOS:** Cada campo requiere de trece argumentos.
   MessageHandler Normalmente 'DROPFALSE

   Coordenada #X
   Coordenada #Y
   Longitud #L
   Altura #h
   Tipo de campo #
   - 1 Campo de texto
   - 3 Campo algebraico
   - 12 Lista de campos
   - 32 Campo Check
   - 44 Campo Choose

   Lista de objetos válidos #
   - MINUSONE Acepta cualquier objeto

   Descompilación
   "Ayuda"
   Dato choose
   Descompilación choose
   Valor de reinicio
   Valor inicial
3. **MESSAGEHANDLER**

   Es un secundario, que se pone en la pila, generalmente se pone 'DROPFALSE.

4. **TÍTULO.**

Es el título que va a ir en la parte superior del formulario, se lo pone en cadena, es decir: “TÍTULO”.

5. **DoInputForm**

**EJEMPLOS:**


: :
“BM:”
BINT1
BINT9
‘DROPFALSE
BINT16
BINT18
BINT130
BINT9
BINT3
MINUSONE
BINT4
“Ingrese cota del BM.”
MINUSONE
MINUSONE
MINUSONE
MINUSONE
BINT1
BINT1
‘DROPFALSE
“NIVELEVV”
DoInputForm



NOT?SEMI
;
@

```
: :
CK0
“f(X):”
BINT0
BINT10
“X1:”
BINT0
BINT19
“Y1:”
BINT70
BINT19
“X2:”
BINT0
BINT28
“Y2:”
BINT70
BINT28
“ ∈”
BINT0
BINT37
“Step/Step”
BINT90
BINT37

‘DROPFALSE
BINT20
BINT8
BINT129
BINT9
BINT23
{
     BINT9
}
BINT4
“Entre o choose ecuación”
“Equations in “
MINUSONE
DUPDUP
‘DROPFALSE
BINT12
BINT17
BINT57
BINT9
BINT3
MINUSONE
```

```
BINT4
“Entre X1.”
MINUSONE
MINUSONE
MINUSONE
MINUSONE

‘DRPFALSE
BINT82
BINT17
BINT57
BINT9
BINT3
MINUSONE
BINT4
“Entre Y1.”
MINUSONE
MINUSONE
MINUSONE
MINUSONE

‘DRPFALSE
BINT12
BINT26
BINT57
BINT9
BINT3
MINUSONE
BINT4
“Entre X2.”
MINUSONE
MINUSONE
MINUSONE
MINUSONE

‘DRPFALSE
BINT82
BINT26
BINT57
BINT9
BINT3
MINUSONE
BINT4
```

```
“Entre Y2.”
MINUSONE
MINUSONE
MINUSONE
MINUSONE

‘DRPFALSE
BINT9
BINT35
BINT20
BINT8
BINT12
MINUSONE
BINT17
“Elija condición.”
{
    {
      “ = “
      BINT1
    }
    {
      “ ≤ “
      BINT2
    }
}
BINT17
{
    “ = “
    BINT1
}
{
      “ = “
      BINT1
}
‘DROPFALSE
BINT31
BINT35
BINT45
BINT9
BINT3
{
    BINT0
}
BINT4
“Entre valor de ∈.”
```

```
MINUSONE
DUP
% .001
DUP

‘DROPFALSE
BINT81
BINT35
BINT7
BINT9
BINT32
MINUSONE
MINUSONE
“Paso a Paso.”
MINUSONE
MINUSONE
FALSE
FALSE


BINT7
BINT8



‘DROPFALSE
“INTERPOLACIÓN LINEAL”
DoInputForm
NOT?SEMI
;
@
```

## 16.- EL BUCLE EXTERNO PARAMETRIZADO (ParOuterLoop)

La palabra ParOuterLoop traducida al español significa Parameterized = Parametrizado, Outer = Externo y Loop = Bucle, que en conjunto significa Bucle Externo Parametrizado.

El ParOuterLoop requiere los siguientes argumentos:

**AppDisplay** Este se evalúa antes de la evaluación de cada tecla. Este debería controlar la actualización de la pantalla que no es controlada por las teclas, y además realizar un especial control de errores.

**AppKeys** Asigna las teclas físicas, en el formato descrito mas adelante.

**NonAppKeyOK?** Una bandera: si es TRUE las teclas físicas no asignadas realizan sus acciones normales. Por lo tanto emiten un sonido (beep).

**DoStdKeys?** Una bandera: si es TRUE entonces la definición de teclas estándar es usada como no-aplicación de las teclas en lugar del valor de proceso de las teclas por defecto.

**AppMenu** Especificación del menú, en el formato descrito mas adelante, o FALSE para dejar el menú actual inalterado.

**#AppMenuPage** La página del menú inicial. Normalmente BINT1 para mostrar la primera pagina.

**SuspenOK?** Una bandera: si es TRUE cualquier comando del usuario que crearía un entorno suspendido y reiniciaría el sistema exterior, en cada vuelta generara un error en cambio.

**ExitCond** Este objeto se evalúa antes de cada actualización de la pantalla y evaluación de la tecla de aplicación. Si es TRUE sale del bucle.

**AppError** El objeto que controla los errores a ser evaluado cuando ocurre un error durante la evaluación de una tecla.

## 16.1 PALABRAS DEL BUCLE EXTERNO PARAMETRIZADO.

El ParOuterLoop esta formado por llamadas (con un apropiado manejo de errores) a las siguientes palabras:

**POLSaveUI** Salva la interfase del usuario en un entorno temporal.

**POLSetUI** Ajusta la interfase actual del usuario, de acuerdo a los parámetros dados.

**POLKeyUI** Muestra, lee y evalúa teclas. Detecta errores y sale según la interfase del usuario especificada por POLSetUI.

**POLRestoreUI** Restaura la interfase del usuario salvada por POLSaveUI y abandona el entrono temporal.

**POLResUI&Err** Restaura la interfase del usuario y errores. Esta es usada cuando hay un error no detectado dentro del ParOuterLoop.

Las palabras de compilación del ParOuterLoop son:

```
: :
POLSaveUI          Salva la interfase actual del usuario
ERRSET             Empieza la captura de errores
: :
        POLSetUI           Ajusta la nueva interfase del usuario
```

```
        PULKeyUI            Detecta las teclas presionadas
;
ERRTRAP
POLResUI&Err                Si hay un error restaura
                            Salva la interfase y error
POLRestoreUI                Restaura la interfase salvada del usuario
;
@
```

NOTA.- El ParOuterLoop crea un entorno temporal cuando salva su interfase del usuario actual, y lo abandona cuando restaura una interfase del usuario salvada.

## 16.2 LA PANTALLA

En el ParOuterLoop el usuario es responsable de preparar la pantalla y ponerla al día, esto si no hay una pantalla por defecto.

La pantalla se pone al día de dos maneras: con el parámetro AppDisplay y mediante la asignación de teclas AppKeys.

## 16.3 DETECTANDO ERRORES

Si ocurre un error durante la ejecución de una tecla, AppError se ejecuta. Este objeto es responsable de procesar cualesquier error mientras ParOuterLoop esta corriendo. AppError debe determinar el error especifico y actuar de acuerdo con él. O usted puede especificar ERRJMP como AppError que quiere decir que su aplicación no maneja ningún error.

## 16.4 ASIGNACIÓN DE TECLAS FÍSICAS

En el ParOuterLoop cualquier tecla en cualquiera de los seis planos básicos puede asignarse una nueva función.

Si una tecla no es asignada por la aplicación, y el NonAppKeyOK? es TRUE, la definición de tecla estándar es ejecutada si el DoStdKeys? es TRUE, o, si se esta disponible, la tecla del usuario es asignada si es FALSE.

Si NonAppKeyOK? es FALSE, se produce un pitido de advertencia, y nada mas es hecho.

La mayoría de tiempo NonAppKeysOK? debe ponerse a FALSE.

El AppKeys es un parámetro secundario, que debe tomar como argumento el código de la tecla (keycode) y el plano (plane), y devuelve cualquiera de la definición de tecla deseado si es TRUE, o FALSE si la aplicación no lo maneja. Específicamente el diagrama de la pila es como sigue:

```
: :
Plano 1
#=casedrop
        : :
        Código tecla 1
        ?CaseKeyDef
                : :
                TakeOver
                Programa a ejecutar
                ;
        .
        .
        .
        Código tecla n
        ?CaseKeyDef
                : :
                TakeOver
                Programa a ejecutar
                ;
        ;
.
.
.
Plano n
#=casedrop
        : :
        Código tecla 1
        ?CaseKeyDef
                : :
                TakeOver
                Programa a ejecutar
                ;
        .
        .
        .
        Código tecla n
        ?CaseKeyDef
                : :
                TakeOver
                Programa a ejecutar
                ;

        ;
.
.
```

.
;
@

## 16.5 ASIGNACIÓN DE TECLAS DE MENÚ.

Usted puede especificar un menú y ser presentado cuando el ParOuterLoop empieza. El formato del parámetro AppMenu es esencialmente el mismo del parámetro ILMenu de InputLine, descrito mas adelante.

La diferencia es que TakeOver no es necesariamente en este caso, puesto que InputLine no es activado.

También desde que las asignaciones de teclas físicas tienen prioridad sobre el menú de tecla asignado, usted debe este código en el parámetro AppKeys, en cada definición de plano:

DUP#<7
Casedropfls

Esto devolverá FALSE cuando una tecla cuyo código es menor que siete (que es uno de las teclas débiles) es presionada. El FALSE forzará la asignación normal para ser corrido, y esta asignación ejecuta la acción definida por AppMenu.

Para el trabajo, el NonAppKeysOK? tiene que ser TRUE,

Ej.:


```
: :
CK0                          No requiere argumentos
FALSE
{
  LAM Exit
}
BIND
‘
: :
  ClrDA1IsStat               Desactiva temporalmente el reloj
  TURNMENUOFF                Desactiva el menú
  GROB 0088A 040003800000000000000000000000000000000000000000000000
  HARDBUFF                   Devuelve el grob de la pantalla actual
  ZEROZERO                   Coordenadas 0,0 del gráfico
  GROB!                      RPL
```

```
;
‘
: :
  BINT1                    Plano de desplazamiento, BINT1: Sin desp.
  #=casedrop
   : :
        DUP#<7
        casedrpfls
        # 2B
        ?CaseKeyDef
        : :
              TakeOver
              “Deflexiones”
              xMSGBOX
        ;
        # 2C
        ?CaseKeyDef
        : :
              TakeOver
              “Distancias”
              xMSGBOX
        ;
        # 2D
        ?CaseKeyDef
        : :
              TakeOver
              “Coordenadas”
              xMSGBOX
        ;
        # 15
        ?CaseKeyDef
        : :
              TakeOver
              TRUE
              ‘
              LAM Exit
              STO
        ;
        DROP
        ‘DoBadKeyT
        ;
        2DROP
        ‘DoBadKey
```

```
  ;
TrueTrue
NULL{}
ONEFALSE
‘
LAM Exit                    Termina condición
‘ERRJMP                     Controla errores
ParOuterLoop                Corre el par outer loop
ClrDAsOK                    Redibuja la pantalla.
;
@
```

## 17.- EL HP48 BROWSER

El HP48 Browser presente aun en la HP49, le permite hacer muchas cosas. Básicamente despliega una lista de entradas a las que usted puede acceder, a diferencia del nuevo HP49 Browser (presente solo en la HP49) que solo le permite seleccionar un solo articulo.

Este “viejo” artefacto tiene unos rasgos que el HP49 I no tiene, como lo es un modo lleno de pantalla. Es sin embrago mas complicado de usar.

El Browser es llamado por ˜Choose. Este retorna los resultados y TRUE o FALSE, dependiendo de la manera como se lo termino, como se vera mas adelante.

A continuación el diagrama de la pila:

(: :Appl $Title: :Converter {}Items Init! resultado TRUE) o
(: :Appl $Title: :Converter {}Items Init! FALSE)

### 17.1 El parámetro ::Appl

Este es un programa que permite la configuración de varios aspectos del Browser. Este funciona como un mensaje handler, que se llama con un numero binario (BINT), que representa el código del mensaje.

A continuación la descripción de algunos de los mensajes:

| Código | Descripción |
|---|---|
| 57 | Número de líneas que el Browser desplegara en la pantalla, el valor por defecto depende del conjunto de caracteres actual y del flag 90 del sistema (→ #). |
| 58 | Altura de la línea del Browser, probablemente este no necesita ser cambiado (→ #). |

59 Ancho de la línea del browser, debe dejarse un espacio para las flechas (en la parte derecha), que indican que hay mas elementos, esto cuando se llena la pagina (→ #).

60 Debe devolverse TRUE, cuando la pantalla esta llena, o FALSE, cuando esta a la mitad. El valor por defecto es a la mitad (FALSE) (→ flag).

61 Deberá devolver TRUE si se permiten objetos chequeados (marcados), o FALSE si no. El valor por defecto es no permitir marcas (→ flag).

62 Retorna el número de elemento, por lo tanto es muy importante manejar este mensaje, cuando cambia el número (→ #).

63 Devuelve las coordenadas de la esquina superior izquierda de la caja de selección Browser. Por lo general no se necesita cambiar este valor por defecto (→ #x #y).

64 Este mensaje devuelve la diferencia entre objetos marcados y el tope de pagina. En efecto, cuando la diferencia es menor que la selección actual y menor del tamaño de la pagina, la calculadora puede causar un error (→ #).

65 Este mensaje es utilizado cuando el fondo necesita ser pintado, es decir, su acción puede usarse cuando se necesita dibujar algo mas en el fondo (→ ).

66 Este mensaje es utilizado cuando se requiere pintar el título. Su acción debe dibujar el titulo en HARDBUF. La mayoría de veces su acción no se maneja, y el título es arrastrado por $Title (→ ).

67 Retorna el título como un grob. La mayoría de las veces este no se usa, y el título es arrastrado por $Title (→ grob).

68 Si el mensaje 67 no esta definido, este es llamado y retorna el título como un grob, pero solo en el modo de pantalla llena (full-screen) (→ grob).

69 Si el mensaje 67 no esta definido, este es llamado y retorna el título como un grob, pero solo en el modo de media pantalla (→ grob).

70 Si el parámetro de $Title no es una cadena nula, esta entrada es llamada para devolver una cadena del título. Esto arrasa el parámetro $Title (→ $).

74 Este mensaje debe dibujar todas las líneas visibles del Browser (→ ).

79 Este mensaje deberá mostrar una línea del Browser. Si esta linea es seleccionada, este mensaje deberá dibujar esta línea en video inversión, o marcarla que es el seleccionado en otra forma (# → ).

80 Este mensaje es una alternativa a proporcionar los artículos como el parámetro items (mas adelante). Proporciona el numero de items y este mensaje deberá retornar el item. Cualquier objeto puede ser retornado ::Converter este comando mandara a convertir esto en una cadena. Si usted quiere tener items con cambios dinámicos, este mensaje le permite, pero el mensaje 82 es mejor en este caso (# → ob).

81 Este mensaje convierte un elemento en un grob. Esto atropella el parámetro ::Converter. Se debe devolver un grob con dimensiones 7NULLLAMx8NULLLAM. Si se habilitan marcas, se debe incorporar estas marcas en el grob si el artículo se verifica (# → grob).

82 Este mensaje es semejante al 80, pero el objeto ya es retornado como una cadena. ::Converter no es llamado solo después. Si este mensaje es utilizado, no se necesita escribir un ::Converter ( #→ $).

83 Retorna una lista que describe el menú. El formato de la lista es similar al del InputLine y al del InptForm (→ {}).

85 Este mensaje es llamado cundo el Browser se empieza, después de que todo ha sido fijo (→ ).

86 Este es llamado cuando un item es chequeado o descartado (# → ).

87 Este mensaje se llama antes de la salida del Browser (→ ).

91 Este mensaje se llama después de que la tecla ON ha sido presionada o la tecla CANCL del menú. Si TRUE es retornado el Browser sale, caso contrario es retornado FALSE, el Browser continua (→ flag).

96 Este mensaje se llama después de que la tecla ENTER ha sido presionada o la tecla OK del menú. Si TRUE es retornado el Browser sale, caso contrario es retornado FALSE, el Browser continua (→ flag).

## 17.2 El parámetro $Title.

Este parámetro especifica el título.

## 17.3 El parámetro ::Converter

Este es un secundario que convierte cualquier tipo de objeto se usa como una lista o una cadena para mostrar. El diagrama de la pila para este objeto es (ob → $).

Si se maneja los mensajes 81 u 82, no se necesita escribir este programa para hacer la conversión. Sin embargo, el Browser le permite al usuario presionar Alpha seguido de una letra para buscar un objeto que empieza con esa letra y salvarlo a él. Esto requiere el parámetro ::Converter aun cuando estos mensajes son suplidos. Debe asegurarse de que estas rutas iguales retorne en cadena.

La entrada de DO>STR puede ser de mucha ayuda aquí.

## 17.4 El parámetro {}Items

Puede especificar una lista de objeto o una lista vacía (NULL{}) y use los mensajes 80, 81 u 82 para proveer los elementos.

## 17.5 El parámetro INIT

Este puede ser un entero binario o una lista. Si es un entero binario 0, el browser trabaja como un visor, desaprobando selecciones. Si es otro entero binario, es el elemento inicialmente seleccionado.

Si se habilitan selecciones múltiples, usted puede especificar una lista de binarios, en cambio representando los elementos inicialmente verificados.

**Ejemplos:**


```
: :
CKO
DOVARS
‘
LAM SM
BINT1
DOBIND
‘
: :
```

```
BINT60
#=casedrop
TrueTrue
BINT62
#=casedrop
: :
        LAM SM
        LENCOMP
        DUP#0=IT
        #1+
        TRUE
;
BINT82
#=casedrop
: :
        LAM SM
        SWAP
        NTHELCOMP
        ITE
        : :
                setStdwid
                FPRT 4 7
        ;
        “No hay SM”
;
BINT83
#=casedrop
: :
        {

                {
                        “AD”
                        : :
                                PushVStack&Clear
                                DoNewMatrix
                                DEPTH
                                #0<>
                                IT
                                : :
                                        LAM SM
                                        SWAP
                                        >TCOMP
                                        ‘
                                        LAM SM
                                        STO
                                        ROMPTR B3 3E
```

```
                ;
                PopMetaVStackDROP
        ;
}
{
        “DEL”
        : :
                LAM SM
                INNERDUP
                #0=case
                DROP
                PushVStack&Keep
                reversym
                DROP
                18GETLAM
                ROLL
                DROP
                18GETLAM
                #1-
                UNROLL
                DEPTH
                {}N
                ‘
                LAM SM
                STO
                PopMetaVStackDROP
                ROMPTR B3 E3
                18GETLAM
                12GETLAM
                #MIN
                18PUTLAM
                FALSE
                ROMPTR B3 19
        ;
}
{
        “EDIT”
        : :
                LAM SM
                18GETLAM




                NTHELCOMP
                NOT?SEMI
```

```
                DoOlMatrix
                NOT?SEMI
                18GETLAM
                LAM SM
                PUTLIST
                ‘
                LAM SM
                STO
        ;
}
{
        “COPY”
        : :
                LAM SM
                LEMCOMP
                #0=case
                DoBadKey
                LAM SM
                DUP
                18GETLAM
                NTHCOMPDROP
                >TCOMP
                ‘
                LAM SM
                STO
                12GETLAM
                #1+
                12PUTLAM
        ;
}
{
        “CLEAR”
        : :
                LAM SM
                LENCOMP
                #0=case
                DoBadKey
                “Borrar todo?”
                AskQuestion
                NOT?SEMI
                NULL{}
                ‘
                LAM SM
                STO
                BINT1
                12PUTLAM
```

```
                                        ;
                                }
                                {
                                        “OK”
                                        FPTR 2 9D
                                }
                        }
                        TRUE
;
        DROPFALSE
;
“MATRICES SM”
‘
NULL: :
NULL{}
BINT1
ROMPTR B3 0
ABND
;
@
```

### 17.6 VARIABLES UASADAS POR EL HP48 BROWSER.

| LAM | DESCRIPCIÓN | TIPO |
|---|---|---|
| 1 | Usada por CACHE | n/a |
| 2 | Condición para terminar POL | flag |
| 3 | Estado de despliegue inicial. Esta es una lista en este formato: { DA1IsStatFlag DA2bEditFlag DA1BadFlag DA2aBadFlag DA2bBadFlag DA3BadFlag } | { } |
| 4 | Menú antes de correr Browser | grob 131x8 |
| 5 | Pantalla antes de correr Browser | grob 131x56 |
| 6 | Desplazamiento en página. | # |
| 7 | Altura de la línea del Browser | # |
| 9 | Coordenada x de la esquina superior izquierda del Browser en el HARDBUFF | # |
| 10 | Coordenada y de la esquina superior izquierda del Browser en el HARDBUFF | # |
| 11 | Tamaño de la página | # |
| 12 | Número de elementos | # |
| 13 | Menú | { } |
| 14 | Full screen? | flag |
| 15 | Lista de indexes de items chequedos | flag |

| | | | |
|---|---|---|---|
| 16 | Marcas de chk habilitados? | | flag |
| 17 | TRUE si es un Browser y FALSE si es un Viewer | flag | |
| 18 | Index actual seleccionado | # | |
| 19 | {}items | | {} |
| 20 | : :Converter | | : : |
| 21 | $Title | | $ |
| 22 | : :Appl | | : : |

# PARTE IV

# ANEXOS

## 18. MENSAJES DE RRROR.

Los siguientes mensajes de error fueron tomados textualmente del documento HP49Gmsg# de la versión 1.17.

# 1: Insufficient Memory
# 2: Directory Recursion
# 3: Undefined Local Name
# 4: Undefined XLIB Name
# 5: Memory Clear
# 6: Power Lost
# 7: Warning:
# 8: Invalid Card Data
# 9: Object In Use
# A: Port Not Available
# B: No Room in Port
# C: Object Not in Port
# D: Recovering Memory
# E: Try To Recover Memory?
# F: Replace RAM, Press ON
# 10: No Mem To Config All
# 11: Undefined FPTR Name
# 12: Invalid Bank Data
# 13: Full Check Bad Crc
# 14: Cmprs: not a user bank
# 15: No or 2 system bank
# 16: Invalid bank
# 17: Invalid bank number
# 18: Inexisting pack
# 19: Pack twice
# 1A: Ins. Mem.
# 1B: Erase Fail, Rom faulty
# 1C: Erase Fail, Low bats
# 1D: Erase Fail, Locked Block
# 1E: Write Adr outside ROM
# 1F: Write Fail, Rom Faulty
# 20: Write Fail, Low bats
# 21: Write Fail, Locked Block
# 101: No Room to Save Stack
# 102: Can't Edit Null Char.
# 103: Invalid User Function
# 104: No Current Equation
# 106: Invalid Syntax
# 107: Real Number
# 108: Complex Number
# 109: String
# 10A: Real Array
# 10B: Complex Array
# 10C: List
# 10D: Global Name
# 10E: Local Name
# 10F: Program
# 110: Algebraic
# 111: Binary Integer
# 112: Graphic
# 113: Tagged
# 114: Unit
# 115: XLIB Name
# 116: Directory
# 117: Library
# 118: Backup
# 119: Function
# 11A: Command
# 11B: System Binary
# 11C: Long Real
# 11D: Long Complex
# 11E: Linked Array
# 11F: Character
# 120: Code
# 121: Library Data
# 122: External
# 124: LAST STACK Disabled
# 125: LAST CMD Disabled
# 126: HALT Not Allowed
# 127: Array

# 128: Wrong Argument Count
# 129: Circular Reference
# 12A: Directory Not Allowed
# 12B: Non-Empty Directory
# 12C: Invalid Definition
# 12D: Missing Library
# 12E: Invalid PPAR
# 12F: Non-Real Result
# 130: Unable to Isolate
# 131: No Room to Show Stack
# 132: Warning:
# 133: Error:
# 134: Purge?
# 135: Out of Memory
# 136: Stack
# 137: Last Stack
# 138: Last Commands
# 139: Key Assignments
# 13A: Alarms
# 13B: Last Arguments
# 13C: Name Conflict
# 13D: Command Line
# 13F: Interrupted
# 140: Integer
# 141: Symbolic Matrix
# 142: Font
# 143: Aplet
# 144: Extended Real
# 145: Extended Complex
# 146: FlashPtr
# 147: Extended Ptr
# 148: MiniFont
# 149: Extended 1
# 14A: Extended 2
# 14B: Extended 3
# 14C: YES
# 14D: NO
# 14E: TRUE
# 14F: FALSE
# 150: Are you sure?
# 151: Low Memory Condition Please Wait...
# 152: CATALOG
# 153: Nonexistent Find Pattern
# 154: Not Found
# 155: Nonexistent Replace Pattern
# 156: Can't Find Selection
# 157: Y= not available
# 158: Warning: Changes will not be saved
# 159: Result not editable in EQW
# 201: Too Few Arguments
# 202: Bad Argument Type
# 203: Bad Argument Value
# 204: Undefined Name
# 205: LASTARG Disabled
# 206: Incomplete Subexpression
# 207: Implicit () off
# 208: Implicit () on
# 301: Positive Underflow
# 302: Negative Underflow
# 303: Overflow
# 304: Undefined Result
# 305: Infinite Result
# 501: Invalid Dimension
# 502: Invalid Array Element
# 503: Deleting Row
# 504: Deleting Column
# 505: Inserting Row
# 506: Inserting Column
# 601: Invalid … Data
# 602: Nonexistent …DAT
# 603: Insufficient … Data
# 604: Invalid …PAR
# 605: Invalid … Data LN(Neg)
# 606: Invalid … Data LN(0)
# 607: Invalid EQ
# 608: Current equation:
# 609: No current equation.
# 60A: Enter eqn, press NEW
# 60B: Name the equation, press ENTER
# 60C: Select plot type
# 60D: Empty catalog
# 60E: undefined
# 60F: No stat data to plot
# 610: Autoscaling
# 611: Solving for
# 612: No current data. Enter
# 613: data point, press …+
# 614: Select a model
# 615: No alarms pending.
# 616: Press ALRM to create
# 617: Next alarm:
# 618: Past due alarm:

# 619: Acknowledged
# 61A: Enter alarm, press SET
# 61B: Select repeat interval
# 61C: I/O setup menu
# 61D: Plot type:
# 61E: ""
# 61F: (OFF SCREEN)
# 620: Invalid PTYPE
# 621: Name the stat data, press ENTER
# 622: Enter value (zoom out if >1), press ENTER
# 623: Copied to stack
# 624: x axis zoom w/AUTO.
# 625: x axis zoom.
# 626: y axis zoom.
# 627: x and y axis zoom.
# 628: IR/wire:
# 629: ASCII/binary:
# 62A: baud:
# 62B: parity:
# 62C: checksum type:
# 62D: translate code:
# 62E: Enter matrix, then NEW
# 62F: No Associated Numeric View
# 701: Algebraic
# 702: RPN
# 703: Standard
# 704: Std
# 705: Fixed
# 706: Fix
# 707: Scientific
# 708: Sci
# 709: Engineering
# 70A: Eng
# 70B: Degrees
# 70C: Radians
# 70D: Grads
# 70E: Rectangular
# 70F: Polar
# 710: Spherical
# 711: Operating Mode
# 712: Number Format
# 713: Angle Measure
# 714: Coord System
# 715: FM,
# 716: Beep
# 717: Key Click
# 718: Last Stack
# 719: Choose calculator operating mode
# 71A: Choose number display format
# 71B: Choose decimal places to display
# 71C: Choose angle measure
# 71D: Choose coordinate system
# 71E: Use comma as fraction mark?
# 71F: Enable standard beep?
# 720: Enable key click?
# 721: Save last stk for UNDO and ANS?
# 722: CALCULATOR MODES
# 723: Font:
# 724: Stack:
# 725: Small
# 726: Textbook
# 727: Edit:
# 728: Small
# 729: Full Page
# 72A: Indent
# 72B: EQW:
# 72C: Small
# 72D: Small Stack Disp
# 72E: Header:
# 72F: Clock
# 730: Analog
# 731: Choose system font
# 732: Display stack using small font?
# 733: Use pretty print in the stack?
# 734: Edit using small font?
# 735: Edit in full page?
# 736: Automatically indent new lines?
# 737: Edit in EQW using small font?
# 738: Display EQW using small font?
# 739: Choose header height
# 73A: Display ticking clock?
# 73B: Analog clock?
# 73C: DISPLAY MODES
# 73D: Indep var:
# 73E: Modulo:
# 73F: Verbose
# 740: Step/Step
# 741: Complex
# 742: Approx
# 743: Incr Pow
# 744: Simp Non-Rational

# 745: Rigorous
# 746: Numeric
# 747: Enter independent variable name
# 748: Enter modulo value
# 749: Display calculus information?
# 74A: Perform operations step by step?
# 74B: Allow complex numbers?
# 74C: Perform approx calculations?
# 74D: Increasing polynomial ordering?
# 74E: Simplify non rational expr?
# 74F: Don't simplify |X| to X?
# 750: Replace constants by values?
# 751: CAS MODES
# 752: Goto row:
# 753: Goto column:
# 754: Specify a row to go to

# 755: Specify a column to go to
# 756: Matrix Writer
# 757: Bad range value
# 758: Start:
# 759: Step:
# 75A: Type:
# 75B: Zoom:
# 75C: Small Font
# 75D: File:
# 75E: Enter starting value
# 75F: Enter increment value
# 760: Choose table format
# 761: Enter zoom factor
# 762: Display table using small font?
# 763: Enter a filename to save data
# 764: TABLE SETUP
# 765: Automatic
# 766: Build Your Own
# 767: Function
# 768: Polar
# 769: Parametric
# 76A: Diff Eq
# 76B: Conic
# 76C: Truth
# 76D: Histogram
# 76E: Bar
# 76F: Scatter
# 770: Slopefield
# 771: Fast3D
# 772: Wireframe
# 773: Ps-Contour
# 774: Y-Slice
# 775: Gridmap
# 776: Pr-Surface
# 777: Deg
# 778: Rad
# 779: Grad
# 77A: Type:
# 77B: €:
# 77C: EQ:
# 77D: Indep:
# 77E: Connect
# 77F: Simult
# 780: H-Tick:
# 781: V-Tick:
# 782: Pixels
# 783: Depnd:
# 784: Save Animation
# 785: …DAT:
# 786: Col:
# 787: Cols:
# 788: F:
# 789: H-Var:
# 78A: V-Var:
# 78B: Stiff
# 78C: ^F^Y:
# 78D: ^F^T:
# 78E: Choose type of plot
# 78F: Choose angle measure
# 790: Enter function(s) to plot
# 791: Enter independent variable name
# 792: Connect plot points?
# 793: Plot functions simultaneously?
# 794: Enter horizontal tick spacing
# 795: Enter vertical tick spacing
# 796: Tick spacing units are pixels?
# 797: Enter dependent variable name
# 798: Save slices animation?
# 799: Enter data to plot
# 79A: Enter col to use for horizontal
# 79B: Enter col to use for vertical
# 79C: Enter horizontal variable

# 79D: Enter vertical variable
# 79E: Use stiff diff eq solver?
# 79F: Enter derivative w.r.t. soln
# 7A0: Enter derivative w.r.t. indep
# 7A1: PLOT SETUP
# 7A2: H-View:
# 7A3: V-View:
# 7A4: Indep Low:
# 7A5: High:
# 7A6: Step:
# 7A7: Pixels
# 7A8: Depnd Low:
# 7A9: High:
# 7AA: X-Left:
# 7AB: X-Right:
# 7AC: Y-Near:
# 7AD: Y-Far:
# 7AE: Step Indep:
# 7AF: Depnd:
# 7B0: Bar Width:
# 7B1: Z-Low:
# 7B2: Z-High:
# 7B3: XE:
# 7B4: YE:
# 7B5: ZE:
# 7B6: Init:
# 7B7: Final:
# 7B8: Init-Soln:
# 7B9: Tol:
# 7BA: XXLeft:
# 7BB: XXRight:
# 7BC: YYNear:
# 7BD: YYFar:
# 7BE: Enter minimum horizontal value
# 7BF: Enter maximum horizontal value
# 7C0: Enter minimum vertical value
# 7C1: Enter maximum vertical value
# 7C2: Enter minimum indep var value
# 7C3: Enter maximum indep var value
# 7C4: Enter indep var increment
# 7C5: Indep step units are pixels?
# 7C6: Enter minimum depend var value
# 7C7: Enter maximum depend var value
# 7C8: Enter bar width
# 7C9: Enter minimum Z view-volume val
# 7CA: Enter maximum Z view-volume val
# 7CB: Enter X eyepoint coordinate
# 7CC: Enter Y eyepoint coordinate
# 7CD: Enter Z eyepoint coordinate
# 7CE: Enter absolute error tolerance
# 7CF: Enter minimum XX range value
# 7D0: Enter maximum XX range value
# 7D1: Enter minimum YY range value
# 7D2: Enter maximum YY range value
# 7D3: PLOT WINDOW
# 7D4: Default
# 7D5: FUNCTION
# 7D6: POLAR
# 7D7: PARAMETRIC
# 7D8: DIFF EQ
# 7D9: CONIC
# 7DA: TRUTH
# 7DB: HISTOGRAM
# 7DC: BAR
# 7DD: SCATTER
# 7DE: SLOPEFIELD
# 7DF: FAST3D
# 7E0: WIREFRAME
# 7E1: PS-CONTOUR
# 7E2: Y-SLICE
# 7E3: GRIDMAP
# 7E4: PR-SURFACE
# 7E5: PLOT WINDOW -
# 7E6: Enter minimum X view-volume val
# 7E7: Enter maximum X view-volume val
# 7E8: Enter minimum Y view-volume val
# 7E9: Enter maximum Y view-volume val
# 7EA: Enter indep var sample count
# 7EB: Enter depnd var sample count
# 7EC: Goto Level:
# 7ED: Specify a level to go to
# 7EE: HISTORY
# 801: Must be >= 0
# 802: Must be bewteen 0 and 1
# 803: μ0:
# 804: □:
# 805: N:
# 806: Œ:
# 807: ˜:
# 808: Null hypothesis population mean
# 809: Sample mean
# 80A: Sample Size

# 80B: Significance level
# 80C: Population standard deviation
# 80D: Z-TEST: 1 µ, KNOWN ˜
# 80E: Alternative Hypothesis
# 80F: □1:
# 810: ˜1:
# 811: N1:
# 812: Œ:
# 813: □2:
# 814: ˜2:
# 815: N2:
# 816: Sample mean for population 1
# 817: Std deviation for population 1
# 818: Sample size for population 1
# 819: Significance level
# 81A: Sample mean for population 2
# 81B: Std deviation for population 2
# 81C: Sample size for population 2
# 81D: Z-TEST: 2 µ, KNOWN ˜
# 81E: ‡0:
# 81F: x:
# 820: N:
# 821: Œ:
# 822: Null hyp. population proportion
# 823: Success count
# 824: Sample size
# 825: Significance level
# 826: Z-TEST: 1 P
# 827: X1:
# 828: N1:
# 829: Œ:
# 82A: X2:
# 82B: N2:
# 82C: Success count for sample 1
# 82D: Size of sample 1
# 82E: Significance level
# 82F: Success count for sample 2
# 830: Size of sample 2
# 831: Z-TEST: 2 P
# 832: □:
# 833: Sx:
# 834: µ0:
# 835: Œ:
# 836: N:
# 837: Null hypothesis population mean
# 838: Sample Standard deviation
# 839: Sample Mean
# 83A: Significance level
# 83B: Sample size
# 83C: T-TEST: 1 µ, UNKNOWN ˜
# 83D: □1:
# 83E: S1:
# 83F: N1:
# 840: Œ:
# 841: □2:
# 842: S2:
# 843: N2:
# 844: Pooled?
# 845: Sample mean for population 1
# 846: Std deviation for sample 1
# 847: Sample size for population 1
# 848: Significance level
# 849: Sample mean for population2
# 84A: Std deviation for sample 2
# 84B: Sample size for population 2
# 84C: "Pooled" if checked
# 84D: T-TEST: 2 µ, UNKNOWN ˜
# 84E: □:
# 84F: ˜:
# 850: N:
# 851: C:
# 852: Sample mean
# 853: Population standard deviation
# 854: Sample size
# 855: Confidence level
# 856: CONF. INT.: 1 µ, KNOWN ˜
# 857: □1:
# 858: ˜1:
# 859: N1:
# 85A: C:
# 85B: □2:
# 85C: ˜2:
# 85D: N2:
# 85E: Sample mean for population 1
# 85F: Std deviation for sample 1
# 860: Size of sample 1
# 861: Sample mean for population 2
# 862: Std deviation for sample 2
# 863: Size of sample 2
# 864: Confidence level
# 865: CONF. INT.: 2 µ, KNOWN ˜
# 866: x:

# 867: N:
# 868: C:
# 869: Sample success count
# 86A: Sample size
# 86B: Confidence level
# 86C: CONF. INT.: 1 P
# 86D: ☐1:
# 86E: N1:
# 86F: C:
# 870: ☐2:
# 871: N2:
# 872: Sample 1 success count
# 873: Sample 1 size
# 874: Sample 2 success count
# 875: Sample 2 size
# 876: Confidence level
# 877: CONF. INT.: 2 P
# 878: ☐:
# 879: Sx:
# 87A: N:
# 87B: C:
# 87C: Sample mean
# 87D: Sample standard deviation
# 87E: Sample size
# 87F: Confidence level
# 880: CONF. INT.: 1 μ, UNKNOWN ˜
# 881: ☐1:
# 882: S1:
# 883: N1:
# 884: C:
# 885: ☐2:
# 886: S2:
# 887: N2:
# 888: Pooled
# 889: Sample 1 mean
# 88A: Std deviation for sample 1
# 88B: Sample 1 size
# 88C: Sample 2 mean
# 88D: Std deviation for sample 2
# 88E: Sample 2 size
# 88F: Confidence level
# 890: Pooled if checked
# 891: CONF. INT.: 2 μ, UNKNOWN ˜
# 892: Search for:
# 893: Replace by:
# 894: Case Sensitive
# 895: Search For:
# 896: Enter search pattern
# 897: Enter replace pattern
# 898: Case sensitive search?
# 899: Enter search pattern
# 89A: FIND REPLACE
# 89B: FIND
# 89C: Goto Line:
# 89D: Specify a line to go to
# 89E: GOTO LINE
# 89F: Goto Position:
# 8A0: Specify a position to go to
# 8A1: GOTO POSITION
# 8A2: H-Factor:
# 8A3: V-Factor:
# 8A4: Recenter on cursor
# 8A5: Enter horizontal zoom factor
# 8A6: Enter vertical zoom factor
# 8A7: Recenter plot on cursor?
# 8A8: ZOOM FACTOR
# 8A9: Object:
# 8AA: Name:
# 8AB: Directory
# 8AC: Enter New Object
# 8AD: Enter variable name
# 8AE: Create a new directory?
# 8AF: NEW VARIABLE
# 8B0: Select Object
# 901: [not shown - very long help text for some statistics app.]
# 902: [not shown - very long help text for some statistics app.]
# 903: [not shown - very long help text for some statistics app.]
# 904: [not shown - very long help text for some statistics app.]
# 905: [not shown - very long help text for some statistics app.]
# 906: [not shown - very long help text for some statistics app.]
# 907: [not shown - very long help text for some statistics app.]
# 908: [not shown - very long help text for some statistics app.]
# 909: [not shown - very long help text for some statistics app.]

# 90A: [not shown - very long help text for some statistics app.]
# 90B: [not shown - very long help text for some statistics app.]
# 90C: [not shown - very long help text for some statistics app.]
# 90D: Inconclusive result
# A01: Bad Guess(es)
# A02: Constant?
# A03: Interrupted
# A04: Zero
# A05: Sign Reversal
# A06: Extremum
# A07: Left
# A08: Right
# A09: Expr
# B01: Invalid Unit
# B02: Inconsistent Units
# C01: Bad Packet Block Check
# C02: Timeout
# C03: Receive Error
# C04: Receive Buffer Overrun
# C05: Parity Error
# C06: Transfer Failed
# C07: Protocol Error
# C08: Invalid Server Cmd.
# C09: Port Closed
# C0A: Connecting
# C0B: Retry #
# C0C: Awaiting Server Cmd.
# C0D: Sending
# C0E: Receiving
# C0F: Object Discarded
# C10: Packet #
# C11: Processing Command
# C12: Invalid IOPAR
# C13: Invalid PRTPAR
# C14: Low Battery
# C15: Empty Stack
# C16: Row
# C17: Invalid Name
# D01: Invalid Date
# D02: Invalid Time
# D03: Invalid Repeat
# D04: Nonexistent Alarm
# B901: Press [CONT] for menu
# B902: reset/delete this field
# B903: Reset value
# B904: Delete value
# B905: Reset all
# B906: Valid object types:
# B907: Valid object type:
# B908: Any object
# B909: Real number
# B90A: (Complex num)
# B90B: "String"
# B90C: [ Real array ]
# B90D: [(Cmpl array)]
# B90E: { List }
# B90F: Name
# B910: « Program »
# B911: 'Algebraic'
# B912: # Binary int
# B913: _Unit object
# B914: Invalid object type
# B915: Invalid object value
# B916: Calculator Modes
# B917: Number Format:
# B918: Angle Measure:
# B919: Coord System:
# B91A: Beep
# B91B: Clock
# B91C: FM,
# B91D: Choose number display format
# B91E: Enter decimal places to display
# B91F: Choose angle measure
# B920: Choose coordinate system
# B921: Enable standard beep?
# B922: Display ticking clock?
# B923: Use comma as fraction mark?
# B924: Standard
# B925: Std
# B926: Fixed
# B927: Fix
# B928: Scientific
# B929: Sci
# B92A: Engineering
# B92B: Eng
# B92C: Degrees
# B92D: Deg
# B92E: Radians
# B92F: Rad

# B930: Grads
# B931: Grad
# B932: Rectangular
# B933: Polar
# B934: Spherical
# B935: SYSTEM FLAGS
# B936: 01 General solutions
# B937: 02 Constant ☐ symb
# B938: 03 Function ☐ symb
# B939: 14 Payment at end
# B93A: 19 ☐V2 ☐ vector
# B93B: 20 Underflow ☐ 0
# B93C: 21 Overflow ☐ ±9E499
# B93D: 22 Infinite ☐ error
# B93E: 27 'X+Y*i' ☐ '(X,Y)'
# B93F: 28 Sequential plot
# B940: 29 Draw axes too
# B941: 31 Connect points
# B942: 32 Solid cursor
# B943: 35 ASCII transfer
# B944: 36 RECV renames
# B945: 37 Single-space prnt
# B946: 38 Add linefeeds
# B947: 39 Show I/O messages
# B948: 40 Don't show clock
# B949: 41 12-hour clock
# B94A: 42 mm/dd/yy format
# B94B: 43 Reschedule alarm
# B94C: 44 Delete alarm
# B94D: 52 Show many lines
# B94E: 53 No extra parens
# B94F: 54 Tiny element ☐ 0
# B950: 55 Save last args
# B951: 57 Alarm beep on
# B952: 58 Show INFO
# B953: 59 Show variables
# B954: 60 [Œ][Œ] locks
# B955: 61 [USR][USR] locks
# B956: 62 User keys off
# B957: 63 Custom ENTER off
# B958: 65 All multiline
# B959: 66 Stk: x lines str
# B95A: 67 Digital clock
# B95B: 68 No autoIndent
# B95C: 69 Line edit
# B95D: 70 ☐GROB 1 line str
# B95E: 71 Show addresses
# B95F: 72 Stack:  cur fnt
# B960: 73 Edit: cur fnt
# B961: 74 Right stack disp
# B962: 76 Purge confirm
# B963: 79 Algebraic stk
# B964: 80 EQW cur stk font
# B965: 81 GRB Alg cur font
# B966: 82 EQW edit cur font
# B967: 83 Display grobs on
# B968: 85 Normal stk disp
# B969: 90 CHOOSE cur font
# B96A: 91 MTRW:matrix
# B96B: 92 MASD asm mode
# B96C: 94 Result = LASTCMD
# B96D: 97 List: horiz disp
# B96E: 98 Vector:horiz disp
# B96F: 99 CAS: quiet
# B970: 100 Step by step off
# B971: 103 Complex off
# B972: 105 Exact mode on
# B973: 106 Simp. in series
# B974: 109 Sym. factorize
# B975: 110 Normal matrices
# B976: 111 Simp non rat.
# B977: 113 Linear simp on
# B978: 114 Disp 1+x ☐ x+1
# B979: 116 Prefer cos()
# B97A: 117 CHOOSE boxes
# B97B: 119 Rigorous on
# B97C: 120 Silent mode off
# B97D: 01 Principal value
# B97E: 02 Constant ☐ num
# B97F: 03 Function ☐ num
# B980: 14 Payment at begin
# B981: 19 ☐V2 ☐ complex
# B982: 20 Underflow ☐ error
# B983: 21 Overflow ☐ error
# B984: 22 Infinite ☐ ±9E499
# B985: 27 'X+Y*i' ☐ 'X+Y*i'
# B986: 28 Simultaneous plot
# B987: 29 Don't draw axes
# B988: 31 Plot points only
# B989: 32 Inverse cursor
# B98A: 35 Binary transfer
# B98B: 36 RECV overwrites

# B98C: 37 Double-space prnt
# B98D: 38 No linefeeds
# B98E: 39 No I/O messages
# B98F: 40 Show clock
# B990: 41 24-hour clock
# B991: 42 dd.mm.yy format
# B992: 43 Don't reschedule
# B993: 44 Save alarm
# B994: 52 Show one line
# B995: 53 Show all parens
# B996: 54 Use tiny element
# B997: 55 No last args
# B998: 57 Alarm beep off
# B999: 58 Don't show INFO
# B99A: 59 Show names only
# B99B: 60 [Œ] locks Alpha
# B99C: 61 [USR] locks User
# B99D: 62 User keys on
# B99E: 63 Custom ENTER on
# B99F: 65 Level 1 multiline
# B9A0: 66 Stk: 1 line str
# B9A1: 67 Analog clock
# B9A2: 68 AutoIndent
# B9A3: 69 Infinite line edit
# B9A4: 70 □GROB x lines str
# B9A5: 71 No addresses
# B9A6: 72 Stack: mini font
# B9A7: 73 Edit: mini font
# B9A8: 74 Left stack disp
# B9A9: 76 No purge confirm
# B9AA: 79 Std stack
# B9AB: 80 EQW mini stk font
# B9AC: 81 GRB Alg mini font
# B9AD: 82 EQW edit mini fnt
# B9AE: 83 Display grobs off
# B9AF: 85 SysRPL stk disp
# B9B0: 90 CHOOSE mini font
# B9B1: 91 MTRW:list of list
# B9B2: 92 MASD SysRPL mode
# B9B3: 94 Result <> LASTCMD
# B9B4: 97 List: vert disp
# B9B5: 98 Vector:vert disp
# B9B6: 99 CAS: verbose
# B9B7: 100 Step by step on
# B9B8: 103 Complex on
# B9B9: 105 Approx. mode on
# B9BA: 106 !Simp. in series
# B9BB: 109 Num factorize
# B9BC: 110 Large matrices
# B9BD: 111 !Simp non rat.
# B9BE: 113 Linear simp off
# B9BF: 114 Disp x+1 □ 1+x
# B9C0: 116 Prefer sin()
# B9C1: 117 Soft MENU
# B9C2: 119 Rigorous off
# B9C3: 120 Silent mode on
# B9C4: Object:
# B9C5: Obs in
# B9C6: Name:
# BA01: 1.Send to HP 49
# BA02: 2.Get from HP 49
# BA03: 3.Print display
# BA04: 4.Print
# BA05: 5.Transfer
# BA06: 6.Start Server
# BA07: Enter names of vars to send
# BA08: Vars in
# BA09: SEND TO HP 49
# BA0A: Port:
# BA0B: Dbl-Space
# BA0C: Delay:
# BA0D: Xlat:
# BA0E: Linef
# BA0F: Baud:
# BA10: Parity:
# BA11: Len:
# BA12: Choose print port
# BA13: Enter object(s) to print
# BA14: Print extra space between lines?
# BA15: Enter delay between lines
# BA16: Choose character translations
# BA17: Print linefeed between lines?
# BA18: Choose baud rate
# BA19: Choose parity
# BA1A: Enter printer line length
# BA1B: PRINT
# BA1C: Type:
# BA1D: OvrW
# BA1E: Fmt:
# BA1F: Chk:
# BA20: Choose transfer port
# BA21: Choose type of transfer

# BA22: Enter names of vars to transfer
# BA23: Choose transfer format
# BA24: Choose checksum type
# BA25: Overwrite existing variables?
# BA26: TRANSFER
# BA27: Local vars
# BA28: Remote PC files
# BA29: Files in
# BA2A: Enter name of dir to change to
# BA2B: Choose Remote Directory
# BA2C: Infrared
# BA2D: IR
# BA2E: Wire
# BA2F: Kermit
# BA30: XModem
# BA31: Odd
# BA32: Even
# BA33: Mark
# BA34: Space
# BA35: Spc
# BA36: ASCII
# BA37: ASC
# BA38: Binary
# BA39: Bin
# BA3A: None
# BA3B: Newline (Ch 10)
# BA3C: Newl
# BA3D: Chr 128-159
# BA3E: □159
# BA3F: □255
# BA40: Chr 128-255
# BA41: One-digit arith
# BA42: Two-digit arith
# BA43: Three-digit CRC
# BB01: 1.Single-var
# BB02: 2.Frequencies
# BB03: 3.Fit data
# BB04: 4.Summary stats
# BB05: SINGLE-VARIABLE STATISTICS
# BB06: …DAT:
# BB07: Type:
# BB08: Mean
# BB09: Std Dev
# BB0A: Variance
# BB0B: Total
# BB0C: Maximum
# BB0D: Minimum
# BB0E: Enter statistical data
# BB0F: Enter variable column
# BB10: Choose statistics type
# BB11: Calculate mean?
# BB12: Calculate standard deviation?
# BB13: Calculate variance?
# BB14: Calculate column total?
# BB15: Calculate column maximum?
# BB16: Calculate column minimum?
# BB17: Sample
# BB18: Population
# BB19: FREQUENCIES
# BB1A: X-Min:
# BB1B: Bin Count:
# BB1C: Bin Width:
# BB1D: Enter minimum first bin X value
# BB1E: Enter number of bins
# BB1F: Enter bin width
# BB20: FIT DATA
# BB21: X-Col:
# BB22: Y-Col:
# BB23: Model:
# BB24: Enter indep column number
# BB25: Enter dependent column number
# BB26: Choose statistical model
# BB27: Correlation
# BB28: Covariance
# BB29: PREDICT VALUES
# BB2A: Y:
# BB2B: Enter indep value or press PRED
# BB2C: Enter dep value or press PRED
# BB2D: SUMMARY STATISTICS
# BB2E: Calculate:
# BB2F: …X
# BB30: …Y
# BB31: …X2
# BB32: …Y2
# BB33: …XY
# BB34: N…
# BB35: Calculate sum of X column?
# BB36: Calculate sum of Y column?
# BB37: Calculate sum of squares of X?
# BB38: Calculate sum of squares of Y?
# BB39: Calculate sum of products?
# BB3A: Calculate number of data points?

# BB3B: Linear Fit
# BB3C: Logarithmic Fit
# BB3D: Exponential Fit
# BB3E: Power Fit
# BB3F: Best Fit
# BB40: 5.Hypoth. tests
# BB41: 6.Conf. interval
# BC01: 1.Browse alarms
# BC02: 2.Set alarm
# BC03: 3.Set time, date
# BC04: SET ALARM
# BC05: Message:
# BC06: Time:
# BC07: Date:
# BC08: Repeat:
# BC09: Enter "message" or « action »
# BC0A: Enter hour
# BC0B: Enter minute
# BC0C: Enter second
# BC0D: Choose AM, PM, or 24-hour time
# BC0E: Enter month
# BC0F: Enter day
# BC10: Enter year
# BC11: Enter alarm repeat multiple
# BC12: Enter alarm repeat unit
# BC13: SET TIME AND DATE
# BC14: Choose date display format
# BC15: Monday
# BC16: Tuesday
# BC17: Wednesday
# BC18: Thursday
# BC19: Friday
# BC1A: Saturday
# BC1B: Sunday
# BC1C: None
# BC1D: AM
# BC1E: PM
# BC1F: 24-hour time
# BC20: 24-hr
# BC21: 1 January
# BC22: 2 February
# BC23: 3 March
# BC24: 4 April
# BC25: 5 May
# BC26: 6 June
# BC27: 7 July
# BC28: 8 August
# BC29: 9 September
# BC2A: 10 October
# BC2B: 11 November
# BC2C: 12 December
# BC2D: Week
# BC2E: Day
# BC2F: Hour
# BC30: Minute
# BC31: Second
# BC32: Weeks
# BC33: Days
# BC34: Hours
# BC35: Minutes
# BC36: Seconds
# BC37: Month/Day/Year
# BC38: M/D/Y
# BC39: Day.Month.Year
# BC3A: D.M.Y
# BC3B: ALARMS
# BD01: 1.Integrate
# BD02: 2.Differentiate
# BD03: 3.Taylor poly
# BD04: 4.Isolate var
# BD05: 5.Solve quad
# BD06: 6.Manip expr
# BD07: INTEGRATE
# BD08: Expr:
# BD09: Var:
# BD0A: Result:
# BD0B: Enter expression
# BD0C: Enter variable name
# BD0D: Enter lower limit
# BD0E: Enter upper limit
# BD0F: Choose result type
# BD10: Choose disp format for accuracy
# BD11: DIFFERENTIATE
# BD12: Value:
# BD13: Enter variable value
# BD14: Expression
# BD15: TAYLOR POLYNOMIAL
# BD16: Order:
# BD17: Enter Taylor polynomial order
# BD18: ISOLATE A VARIABLE
# BD19: Principal
# BD1A: Get principal solution only?

# BD1B: SOLVE QUADRATIC
# BD1C: MANIPULATE EXPRESSION
# BD1D: MATCH EXPRESSION
# BD1E: Pattern:
# BD1F: Replacement:
# BD20: Subexpr First
# BD21: Cond:
# BD22: Enter pattern to search for
# BD23: Enter replacement object
# BD24: Search subexpressions first?
# BD25: Enter conditional expression
# BD26: Symbolic
# BD27: Numeric
# BE01: Plot
# BE02: Type:
# BE03: €:
# BE04: H-View:
# BE05: Autoscale
# BE06: V-View:
# BE07: Choose type of plot
# BE08: Choose angle measure
# BE09: Enter function(s) to plot
# BE0A: Enter minimum horizontal value
# BE0B: Enter maximum horizontal value
# BE0C: Autoscale vertical plot range?
# BE0D: Enter minimum vertical value
# BE0E: Enter maximum vertical value
# BE0F: Plot (x(t), y(t))
# BE10: Enter complex-valued func(s)
# BE11: Plot y'(t)=f(t,y)
# BE12: Enter function of INDEP and SOLN
# BE13: Enter derivative w.r.t. SOLN
# BE14: Enter derivative w.r.t. INDEP
# BE15: Use Stiff diff eq solver?
# BE16: …Dat:
# BE17: Col:
# BE18: Wid:
# BE19: Enter data to plot
# BE1A: Arrays in
# BE1B: Enter column to plot
# BE1C: Enter bar width
# BE1D: Cols:
# BE1E: Enter col to use for horizontal
# BE1F: Enter col to use for vertical
# BE20: Steps:
# BE21: Enter indep var sample count
# BE22: Enter dep var sample count
# BE23: Plot Options
# BE24: Lo:
# BE25: Hi:
# BE26: Axes
# BE27: Simult
# BE28: Connect
# BE29: Pixels
# BE2A: H-Tick:
# BE2B: V-Tick:
# BE2C: Enter minimum indep var value
# BE2D: Enter maximum indep var value
# BE2E: Draw axes before plotting?
# BE2F: Connect plot points?
# BE30: Plot functions simultaneously?
# BE31: Enter indep var increment
# BE32: Indep step units are pixels?
# BE33: Enter horizontal tick spacing
# BE34: Enter vertical tick spacing
# BE35: Tick spacing units are pixels?
# BE36: Depnd:
# BE37: Enter dependent var name
# BE38: Enter minimum dep var value
# BE39: Enter maximum dep var value
# BE3A: H-Var:
# BE3B: V-Var:
# BE3C: Enter max indep var increment
# BE3D: Choose horizontal variable
# BE3E: Choose vertical variable
# BE3F: 0 INDEP
# BE40: 1 SOLN
# BE41:  SOLN(
# BE42: X-Left:
# BE43: X-Right:
# BE44: Y-Near:
# BE45: Y-Far:
# BE46: Z-Low:
# BE47: Z-High:
# BE48: Enter minimum X view-volume val
# BE49: Enter maximum X view-volume val
# BE4A: Enter minimum Y view-volume val
# BE4B: Enter maximum Y view-volume val
# BE4C: Enter minimum Z view-volume val
# BE4D: Enter maximum Z view-volume val
# BE4E: XE:
# BE4F: YE:

# BE50: ZE:
# BE51: Enter X eyepoint coordinate
# BE52: Enter Y eyepoint coordinate
# BE53: Enter Z eyepoint coordinate
# BE54: Save Animation
# BE55: Save animation data after plot?
# BE56: XX-Left:
# BE57: XX-Rght:
# BE58: YY-Near:
# BE59: YY-Far:
# BE5A: Enter minimum XX range value
# BE5B: Enter maximum XX range value
# BE5C: Enter minimum YY range value
# BE5D: Enter maximum YY range value
# BE5E: XX and YY Plot Options
# BE5F: Zoom Factors
# BE60: H-Factor:
# BE61: V-Factor:
# BE62: Recenter at Crosshairs
# BE63: Enter horizontal zoom factor
# BE64: Enter vertical zoom factor
# BE65: Recenter plot at crosshairs?
# BE66: Reset plot
# BE67: Dflt
# BE68: Auto
# BE69: Function
# BE6A: Polar
# BE6B: Conic
# BE6C: Truth
# BE6D: Parametric
# BE6E: Diff Eq
# BE6F: Histogram
# BE70: Bar
# BE71: Scatter
# BE72: Slopefield
# BE73: Wireframe
# BE74: Ps-Contour
# BE75: Y-Slice
# BE76: Gridmap
# BE77: Pr-Surface
# BF01: 1.Solve equation
# BF02: 2.Solve diff eq
# BF03: 3.Solve poly
# BF04: 4.Solve lin sys
# BF05: 5.Solve finance
# BF06: SOLVE EQUATION
# BF07: Enter value or press SOLVE
# BF08: Eq:
# BF09: Enter function to solve
# BF0A: Funcs in
# BF0B: Solver Variable Order
# BF0C: Variables:
# BF0D: Enter order of vars to display
# BF0E: SOLVE Y'(T)=F(T,Y)
# BF0F: f:
# BF10: ^f^y:
# BF11: ^f^t:
# BF12: Indep:
# BF13: Init:
# BF14: Final:
# BF15: Soln:
# BF16: Tol:
# BF17: Step:
# BF18: Stiff
# BF19: Enter function of INDEP and SOLN
# BF1A: Enter derivative w.r.t. SOLN
# BF1B: Enter derivative w.r.t. INDEP
# BF1C: Enter independent var name
# BF1D: Enter initial indep var value
# BF1E: Enter final indep var value
# BF1F: Enter solution var name
# BF20: Enter initial solution var value
# BF21: Press SOLVE for final soln value
# BF22: Enter absolute error tolerance
# BF23: Enter initial step size
# BF24: Calculate stiff differential?
# BF25: f
# BF26: Tolerance
# BF27: Solution
# BF28: SOLVE AN·X^N++A1·X+A0
# BF29: Coefficients [ an  a1 a0 ]:
# BF2A: Roots:
# BF2B: Enter coefficients or press SOLVE
# BF2C: Enter roots or press SOLVE
# BF2D: Coefficients
# BF2E: Roots
# BF2F: SOLVE SYSTEM A·X=B
# BF30: A:
# BF31: B:
# BF32: X:
# BF33: Enter coefficients matrix A
# BF34: Enter constants or press SOLVE

# BF35: Enter solutions or press SOLVE
# BF36: Constants
# BF37: Solutions
# BF38: N:
# BF39: I%YR:
# BF3A: PV:
# BF3B: PMT:
# BF3C: P/YR:
# BF3D: FV:
# BF3E: Enter no. of payments or SOLVE
# BF3F: Enter yearly int rate or SOLVE
# BF40: Enter present value or SOLVE
# BF41: Enter payment amount or SOLVE
# BF42: Enter no. of payments per year
# BF43: Enter future value or SOLVE
# BF44: Choose when payments are made
# BF45: TIME VALUE OF MONEY
# BF46: N
# BF47: I%/YR
# BF48: PV
# BF49: PMT
# BF4A: FV
# BF4B: End
# BF4C: Begin
# BF4D: Beg
# BF4E: AMORTIZE
# BF4F: Payments:
# BF50: Principal:
# BF51: Interest:
# BF52: Balance:
# BF53: Enter no. of payments to amort
# BF54: Principal
# BF55: Interest
# BF56: Balance
# C001: Unable to find root
# DE01:  denominator(s)
# DE02: root(s)
# DE03: last
# DE04: obvious
# DE05: factorizing
# DE06: value
# DE07:  test(s)
# DE08: searching
# DE09: TAYLR of □ at
# DE0A: nth
# DE0B:  is
# DE0C:  numerator(s)
# DE0D: Less than
# DE0E: multiplicity
# DE0F: list of
# DE10:  at
# DE11: factor(s)
# DE12: Eigenvalues
# DE13: Computing for
# DE14: Root mult <
# DE15: Numerical to symbolic
# DE16: Invalid operator
# DE17: Result:
# DE18: Pivots
# DE19: Press CONT to go on
# DE1A: Test
# DE1B: To be implemented
# DE1C: Unable to factor
# DE1D: Z is not = 1 mod 4
# DE1E: Z is not prime
# DE1F: Empty {} of equations
# DE20: Not reducible to a rational expression
# DE21: Non unary operator
# DE22: User function
# DE23: Non isolable operator
# DE24: Not exact system
# DE25: Parameters not allowed
# DE26: CAS internal error
# DE27: Invalid ^ for SERIES
# DE28: Operator not implemented (SERIES)
# DE29: No variable in expr.
# DE2A: No solution found
# DE2B: Invalid derivation arg
# DE2C: No solution in ring
# DE2D: Not a linear system
# DE2E: Can't derive int. var
# DE2F: Diff equation order>2
# DE30: INT:invalid var change
# DE31: Mode switch cancelled
# DE32: No name in expression
# DE33: Invalid user function
# DE34: Can't find ODE type
# DE35: Integer too large
# DE36: Unable to find sign
# DE37: Non-symmetric matrix

# DE38: ATAN insufficent order
# DE39: ASIN at infinity undef
# DE3A: Unsigned inf error
# DE3B: LN[Var] comparison err
# DE3C: Undef limit for var
# DE3D: Bounded var error
# DE3E: Got expr. indep of var
# DE3F: Can't state remainder
# DE40: LN of neg argument
# DE41: Insufficient order
# DE42: ABS of non-signed 0
# DE43: Numeric input
# DE44: Singularity! Continue?
# DE45: Cancelled
# DE46: Negative integer
# DE47: Parameter is cur. var. dependent
# DE48: Unsimplified sqrt
# DE49: Non polynomial system
# DE4A: Unable to solve ODE
# DE4B: Array dimension too large
# DE4C: Unable to reduce system
# DE4D: Complex number not allowed
# DE4E: Polyn. valuation must be 0
# DE4F: Mode switch not allowed here
# DE50: Non algebraic in expression
# DE51: Purge current variable
# DE52: Reduction result
# E101: Avogadro's number
# E102: Boltzmann
# E103: molar volume
# E104: universal gas
# E105: std temperature
# E106: std pressure
# E107: Stefan-Boltzmann
# E108: speed of light
# E109: permittivity
# E10A: permeability
# E10B: accel of gravity
# E10C: gravitation
# E10D: Planck's
# E10E: Dirac's
# E10F: electronic charge
# E110: electron mass
# E111: q/me ratio
# E112: proton mass
# E113: mp/me ratio
# E114: fine structure
# E115: mag flux quantum
# E116: Faraday
# E117: Rydberg
# E118: Bohr radius
# E119: Bohr magneton
# E11A: nuclear magneton
# E11B: photon wavelength
# E11C: photon frequency
# E11D: Compton wavelen
# E11E: 1 radian
# E11F: 2‡ radians
# E120: € in trig mode
# E121: Wien's
# E122: k/q
# E123: "0/q
# E124: q*"0
# E125: dielectric const
# E126: SiO2 dielec cons
# E127: ref intensity
# E128: CONSTANTS LIBRARY
# E129: Undefined Constant
# E401: Invalid Mpar
# E402: Single Equation
# E403: EQ Invalid for MINIT
# E404: Too Many Unknowns
# E405: All Variables Known
# E406: Illegal During MROOT
# E407: Solving for
# E408: Searching
# E601: No Solution
# E602: Many or No Solutions
# E603: I%YR/PYR ‰ -100
# E604: Invalid N
# E605: Invalid PYR
# E606: Invalid #Periods
# E607: Undefined TVM Variable
# E608: END mode
# E609: BEGIN mode
# E60A:  payments/year
# E60B: Principal
# E60C: Interest
# E60D: Balance
# E701: NEAR
# E702:  MINE
# E703:  MINES

# E704:  SCORE:
# E705: YOU MADE IT!!
# E706: YOU BLEW UP!!
# 10001: Invalid $ROMID
# 10002: Invalid $TITLE
# 10003: Invalid $MESSAGE
# 10004: Invalid $VISIBLE
# 10005: Invalid $HIDDEN
# 10006: Invalid $EXTPRG
# 10101: Invalid File
# 10102: Too Many
# 10103: Unknow Instruction
# 10104: Bad Field
# 10105: Codage 0-15 expected
# 10106: Codage 1-16 expected
# 10107: Label Expected
# 10108: Hexa Expected
# 10109: Decimal Expected
# 1010A: Can't Find
# 1010B: Label already defined
# 1010C: { expected
# 1010D: } expected
# 1010E: ( expected
# 1010F: Forbidden
# 10110: Bad Expression
# 10111: Jump too Long
# 10112: 1-8 expected
# 10113: Insufisant Memory
# 31401: No Message here