Talent

EXTENSIONES

MSX LOGO

Producido por TELEMATICA S.A. en la Provincia de San Luis - Argentina

Retrocomputing

EXTENSIONES AL TALENT MSX LOGO

MANUAL DE USO

Producido en Argentina TELEMATICA S.A.

Redacción:
Hugo D. Caro

I.S.B.N. 950-9688-11-6

IMPORTANTE: La presente carpeta incluye un manual y un diskette que contiene las Extensiones al MSX-Logo de Talent. Para su utilización se requiere además de la computadora personal Talent, el cartucho del MSX-LOGO de Talent y una unidad de discos flexibles Talent-MSX.
Estas extensiones fueron desarrolladas exclusivamente para MSX - LOGO de Talent y no funcionan con ningún otro Logo desarrollado para MSX.

Hecho el depósito que marca la ley 11.723
Impreso en Argentina
Printed in Argentina

Prefacio

La obtención de un buen producto computacional es el resultado al que se arriba cuando en su realización interviene un equipo de personas que aportan lo mejor de su conocimiento para el logro de los objetivos propuestos. Este es dicho caso.

En el diseño y realización de estas Extensiones al Logo confluye el esfuerzo creativo de un grupo de calificados especialistas quienes tomaron cada uno de ellos el desarrollo de una parte del proyecto.

Así tenemos al Ing. Reggini como autor de las extensiones musicales y el logo tridimensional.

Carlos Ranalli, integrante del plantel de Telemática, tuvo a su cargo el desarrollo de las primitivas de manejo de archivo en disco y copiado de pantalla en impresora.

Y por último, Hugo Caro se encargó de la redacción del presente manual, y de los ejemplos de aplicación de los entornos dicologo y prelogo, sobre ideas del Lic. Miguel Figini.

El producto que ponemos en sus manos es de excelente calidad y no dudamos que le será de suma utilidad. A lo largo del manual encontrará la información necesaria para la obtención del máximo provecho

José A. Moncada
Dto.de Asistencia
al Usuario

Indice

Introducción

Las extensiones al Talent MSX LOGO forman parte de una filosofía de tareas que Talent, por intermedio de Telemática S.A., ha implementado y que consiste en dar un máximo apoyo a todos los productos que comercializa.

En este caso estamos brindando una ampliación al Talent MSX LOGO que incorpora algunas facilidades que el cartucho no posee, y que son útiles para dar un máximo aprovechamiento al resto de los periféricos, como por ejemplo la unidad de discos o una impresora.

Las extensiones que se incorporan están agrupadas en 3 tipos defunciones:

a) Manejo de archivos y copiar pantalla por impresora.
b) Sistema musical LOGO.
c) LOGO tridimensional.

En el presente manual se hace una explicación exhaustiva de todo el entorno de trabajo de cada una de estas nuevas funciones y se acompaña con una aplicación que demuestra su funcionamiento.

Esperamos que puedan aprovechar aún más su Talent MSX LOGO a través de las extensiones que aquí se brindan.

Modo de Uso

Las extensiones al Talent MSX LOGO vienen almacenadas en un disco que acompaña a este manual y que contiene los siguientes archivos:

PRIMIT.OEM
MUSICA
TRI
TRIAUX
BREVES

Por otra parte, los ejemplos que aquí se muestran están incorporados en los siguientes archivos:

DICOLOGO (Ejemplo de manejo de archivos)
PRELOGO (Aplicación del uso de la impresora)
MOZART (Programa generado por el sistema musical)
CUMPLE (Programa generado por el sistema musical)

Todos los archivos se cargan utilizando la primitiva recordar salvo lasque se incorporan en el archivo PRIMIT.OEM, que se cargan automáticamente si al encender el equipo, el disco con las extensiones se encuentra colocado en la unidad de discos y en posición de lectura/grabación.

Hay que destacar que de los tres entornos, dos han sido desarrollados en LOGO, lo que permite ver la gran capacidad de ampliación que posee este lenguaje.

Sin embargo, las extensiones que se incluyen en el archivo PRIMIT.OEM están escritas en código de máquina y no están accesibles para el usuario.

IMPORTANTE:
La configuración mínima necesaria de equipo es la siguiente:

1) Consola
2) Televisor o monitor
3) Unidad de discos ,
4) Impresora (opcional para utilizar la copia de pantallas).

Dadas sus características técnicas, las primitivas integrantes del archivo PRIMIT.OEM (Manejo de archivos y copiado de pantalla) no operan en la Talent Minilan.

El MSX-LOGO y el manejo de archivos.

Todo lenguaje debe proveer diversos procedimientos o comandos que permitan la interacción de la computadora con diversos periféricos, como pueden ser una impresora, la unidad de discos o un grabador de cassettes.

MSX-LOGO no escapa a esta norma y provee diversas posibilidades para manejar los periféricos antes nombrados, mediante la definición de primitivas que vienen incorporadas en el mismo.

Nos referiremos ahora a la interacción del MSX-LOGO con la unidad de discos, pues es la que más ventaja saca a los métodos de almacenamiento auxiliar.

El lenguaje LOGO puede manejar distintos tipos de archivos, que nos permiten almacenar información de programas, datos, etc., en una memoria auxiliar representada por la unidad de discos y su lugar físico de almacenamiento: el diskette.

En esta "memoria auxiliar" se pueden almacenar y recuperar programas, para ejecutarlos o modificarlos, o bien acceder a datos que fueron escritos por un programa.

Al ser MSX-LOGO un lenguaje que no está orientado a realizar tareas de gestión, no disponíamos hasta el momento de primitivas capaces de crear y manejar archivos de datos, pero gracias a las extensiones que aquí describimos, ya disponemos de dichas primitivas.

TIPOS DE ARCHIVOS

Los distintos tipos de archivos a los que se puede acceder desde MSX-LOGO son los que describimos a continuación, indicando en cada caso si las primitivas que permiten su manejo vienen incorporadas en el MSX-LOGO (primitivas) o se ha incorporado al mismo mediante el disco de extensiones (extensiones).

* *Directorio. (primitivas)*

Este "archivo" es en realidad una tabla almacenada en el disco que nos permite saber, en cada momento, los archivos que tenemos almacenados en un determinado diskette. Este "archivo"

lo crea y maneja el sistema Operativo y no se puede modificar desde MSX-LOGO, si bien sí se puede conocer su contenido mediante un comando MSX-LOGO. El comando incorporado en el MSX-LOGO se denomina "dir" y está disponible para todo usuario que disponga de al menos una unidad de disco.

* *Programas (primitivas)*

El manual de uso del MSX-LOGO describe cómo se podía almacenar el "área de trabajo" en un archivo, mediante el comando "guardar", y recuperarlo, más tarde, con el comando "recordar". Hay que recordar que al almacenar el "área de trabajo" lo que se guarda en el archivo son todos los procedimientos, nombres y propiedades que estén definidos al momento de ejecutar el comando "guardar".

* *De pantalla (primitivas)*

Este tipo de archivos almacenan una imagen de los gráficos y textos que se mostraban cuando se ejecutó el comando correspondiente. Los comandos utilizados por MSX-LOGO son "guardardib" para grabar una pantalla en diskette y "recordardib" para recuperar una pantalla.

* *De textos (extensiones)*

Los archivos de texto o "dribble" contienen una copia de los textos que aparecen por pantalla. Permiten obtener un registro de las interacciones de un usuario con la computadora, es decir, los textos que el usuario escribe, y lo que le contesta la computadora. Mediante las extensiones la posibilidad de generar este tipo de archivos ha sido incorporada al MSX-LOGO. Se han implementado dos primitivas "ad-hoc":
Por ejemplo:

```
concopia "grupo3
```

crea un archivo "de textos" llamado "grupo3". A partir de ese momento, todo el diálogo entre el usuario y la computadora se almacenará en dicho archivo. Para que deje de grabar el texto hay que desactivarlo con el comando "sincopia". De esta forma, un profesor puede tener un control sobre los alumnos, al conocer la

historia de lo que han realizado en la computadora, sus errores y aciertos.

Asimismo, el MSX-LOGO trae incorporado para los usuarios que posean impresora una forma de generar archivos "dribble" sobre la misma: con los comandos "conimpresora" y "sinimpresora". Estos comandos no requieren del disco de extensiones, ya que se hallan incorporados en el MSX-LOGO y ni siquiera es necesario para utilizarlos contar con una unidad de discos.

* *De datos (extensiones)*

Los archivos de datos permiten almacenar información, acceder a ella, ya sea en su totalidad o a algún dato concreto, leerlos o modificarlos. Cuando el número de datos con el que trabajamos es pequeño, basta con almacenarlos en variables del "área de trabajo" que denominamos "nombres", ya que al finalizar el trabajo, quedan grabados con sus contenidos finales cuando se guardan los procedimientos en la forma acostumbrada.

Sin embargo, cuando se necesita manejar una cantidad grande de datos se hace menester utilizar a la unidad de discos como una "memoria" suplementaria donde almacenaremos los datos que de otra forma sería imposible procesar. Podemos pensar en los archivos con datos como una "expansión" del área de trabajo que se genera en forma virtual, con la consiguiente ventaja.

Por ejemplo, podemos desear almacenar los datos personales de los alumnos de una escuela, o crear un diccionario que por un lado contenga las definiciones de las palabras incluidas y por otro los sinónimos de dichas palabras.

Manejo de archivos: Creando un diccionario.

Como no hay mejor maestro que la práctica vamos a crear un entorno de trabajo donde se pueda ver claramente cómo se pueden aplicar las extensiones para manejo de archivos en el Talent MSX LOGO. En este caso, damos por conocidas todas las primitivas comunes del Talent MSXLOGO.

Crearemos un diccionario en donde usaremos como medio de almacenamiento el disco en vez de la memoria. Como ejemplo podríamos crear un diccionario castellano-inglés; pero ya se notará que en realidad se puede crear cualquier tipo de diccionario con solo cambiar el archivo en uso (la base de datos).

Comencemos, pues, a describir nuestro diccionario que llamaremos "dicologo".

Lo primero que hay que hacer para poder utilizar un archivo es abrirlo. Veamos el siguiente procedimiento, denominado inicia pues se encarga de iniciar las tareas de nuestro dicologo:

```
para inicia
esc [Nombre archivo a utilizar?]
hacer "nombrear primero 11
si vacia? :nombrear [parar] [inicializar :nombrear]
fin
```

Hasta el momento no hemos visto nada novedoso, así que veamos el procedimiento inicializar: él nos mostrará las novedades.

```
para inicializar :nombrear
abrir :nombrear
fleer :nombrear
fesc :nombrear
fin
```

En este procedimiento vemos tres primitivas nuevas:

```
abrir
fleer
fesc
```

Con estas primitivas ya establecemos la apertura y los modos de trabajo del archivo que vamos a utilizar. La primitiva abrir es la que establece el primer vínculo, es decir, coloca "en línea" el archivo indicado. Cuando se ejecuta la primitiva lo que hace es buscar en el directorio el archivo en cuestión y se posiciona en él, si existe. Si no existe, lo crea.

Las dos primitivas que siguen indican el "modo" en el que se utilizará el archivo.

Con **fleer** (fijarleer), se le indica al LOGO que se desea utilizar el archivo para lectura, o sea, que los datos circularán desde el diskette hacia la computadora. Cuando el archivo ya existe, el sistema se posiciona en el primer caracter que se encuentra disponible en el archivo.

En cambio **fesc** (fijarescribir) selecciona el modo de escritura del archivo en cuestión, o sea, los datos circularán desde la computadora hacia el diskette. Cuando el archivo que se indica ya existe, **fesc** se posiciona sobre el caracter que sigue al último del archivo.

De esta manera es posible continuar escribiendo el archivo sin perder los datos anteriores. Si el archivo no existe, cuando **abrir** lo crea permite que **fesc** se posicione sobre el principio del archivo.

Una característica destacable de las primitivas **fleer** y **fesc** es su independencia respecto de la cantidad de archivos abiertos.

Dado que los comandos de lectura y escritura se refieren siempre a los archivos que se encuentran "on-line", o sea, disponibles para esas tareas, sólo se puede tener un archivo en modo de escritura y otro en modo de lectura (que puede ser el mismo archivo).

Esta particularidad es interesante ya que hace innecesario tener que cerrar y abrir los archivos para poder posicionarse y escribir. Siempre hay que recordar que el último **fesc** o **fleer** es el que está en ese momento activo. Tomemos el siguiente ejemplo :

```
abrir "pepe
abrir "cacho
```

Si ingresamos los siguientes comandos:

```
fesc "pepe
fesc "cacho
```

El archivo que queda en modo escritura es **"cacho** y no **"pepe**. Esto significa que cuando utilicemos los comandos para enviar los datos a estos archivos, éstos se grabarán en el archivo **"cacho**, quedando **"pepe** desafectado hasta que se lo reactive (con **fesc**).

IMPORTANTE:

En todos los casos, el nombre del archivo que acompaña a estas tres primitivas debe ser una palabra (no puede ser una lista).

Ahora ya tenemos el archivo en funcionamiento. Procedamos a cargar nuestro diccionario. Para lograr este objetivo crearemos el procedimiento **agrega**.

```
para agrega
esc [quieres agregar algo mas? si - no]
si 11 = [no] [parar]
esc [escribe la palabra y su relación]
escribir.a 11
agrega
fin
```

La novedad destacable en este procedimiento es el uso de la primitiva **escribir.a**. Esta primitiva funciona de manera totalmente análoga a la primitiva **escribir** del MSX LOGO, con la diferencia que los datos a escribir son enviados al archivo que se encuentre en este momento en modo de escritura (**fesc**).

En nuestro caso nos encargamos de enviar una lista al archivo en cuestión. Carguemos los siguientes datos de muestra:

```
perro dog
gato cat
casa house
mesa table
silla chair
```

En nuestro caso, hemos creado un archivo que podremos considerar como una gran lista que contiene las sublistas [perro dog] [gato cat] [casa house] ... etc.

Todo diccionario que se precie nos permitirá buscar las definiciones que contiene. El nuestro para no ser menos lo implementa de la siguiente manera:

```
para buscar
esc [Que palabra quieres buscar?]
hacer "palabra primero 11
fposleer 0
busqueda
fin
```

Que es el procedimiento que toma la palabra a buscar, mientras que búsqueda realiza la tarea de la siguiente manera:

```
para busqueda
si leerfda? [parar] [hacer "dic 11.a]
si :palabra = primero :dic [esc :dic parar]
busqueda
fin
```

Para pedir la búsqueda de una palabra, se debe ingresar el comando buscar. Por ejemplo, busquemos cómo se dice "gato" en inglés:

```
buscar
Que palabra quieres buscar?
gato
gato cat
?
```

Analicemos las nuevas primitivas utilizadas. La primera que vemos es **`fposleer`**. Significa "fijarposiciónleer" o sea que nos permite ubicarnos sobre el archivo en una posición determinada. Llegados a este punto, tendremos que aclarar algunos temas para poder indicar a qué posición nos referimos.

Cuando accedemos a un archivo, existen varias formas de hacerlo. En nuestro caso, nos referiremos al acceso tipo *secuencial* y al tipo *random* o *acceso aleatorio*. Cuando accedemos a Un archivo en forma secuencial leemos los datos del archivo uno tras Otro, partiendo del principio y siguiendo de esta manera, uno tras otro, hasta el final. Este método de acceso nos recuerda el acceso a datos almacenados en cassette: no se puede acceder a un dato determinado sin pasar antes por los que lo preceden. Las ventajas de este tipo de acceso son que la lectura es bastante rápida y que los grupos de datos (registros) pueden tener longitud variable.

Por otra parte, cuando accedemos a un archivo en forma aleatoria o random, podemos llegar directamente al dato que nos interesa, pero con la condición de que los grupos de datos tengan todos la misma longitud. Este tipo de acceso nos recuerda a un disco musical, en donde uno puede buscar directamente, con la púa, el tema que deseamos escuchar.

En nuestro caso, usaremos un acceso secuencial para la búsqueda de datos. Por lo tanto, para comenzar a buscar desde el principio, lo primero que tenemos que hacer es fijar la posición de lectura en el principio, o sea en la posición 0. Ahora bien, si quisiéramos posicionarnos en la posición 23, dónde estamos parados realmente? En la lista No. 23, en la palabra No. 23 o en el carácter No. 23? Como el sistema no puede conocer "prima facie" qué tipo de datos hay almacenados, lo único seguro es referirse a caracteres. En nuestro caso, fijar la posición de lectura en la posición 23 significa que cuando leamos el siguiente carácter del archivo, será el número 24 (el primero es el número 0).

Luego de esta disquisición, continuemos. Como ya dijimos, búsqueda se encarga de buscar la palabra que contiene :palabra dentro del

diccionario. Para poder rescatar datos del archivo que tiene el diccionario, hemos utilizado el procedimiento **11.a** o sea "leerlista.archivo". Esta nueva primitiva trabaja de manera totalmente análoga que la primitiva **11** del Talent MSX LOGO, pero la lista la obtiene desde el archivo.

Por supuesto, los datos almacenados deben tener forma de lista, pues silos delimitadores no coinciden con los esperados, se pueden obtener resultados que no son los deseados.

En nuestro caso, el primer elemento de la lista almacenada es la clave de búsqueda dentro del diccionario. Pero no podemos buscar eternamente dentro del archivo, ya que éste tiene longitud finita. Para evitar que**11.a** nos avise que no hay más datos que buscar usamos otra primitiva que nos devuelve **cierto** o **falso** indicando si se llegó al fin del archivo: "**leerfda?**" (o sea leerfindearchivo?).Con este sencillo control logramos nuestro objetivo.

Hasta el momento, ya estamos en condiciones de crear un diccionario, agregar datos y buscarlos dentro del mismo.

Sería interesante también poder ver el contenido de todo el diccionario para controlar lo que se ha incorporado al mismo hasta el momento. Para eso utilizamos el procedimiento **muestra.todo**:

```
para muestra.todo
fposleer 0
listar
fin
para listar
si leerfda? [parar]
hacer "ver 11.a
si no vacia? :ver [esc :ver]
listar
fin
```

Nos restarían tres funciones básicas: ordenar, editar (modificar) y borrar.

De estas tres funciones, la única que implementaremos es la de borrar, pues siempre se puede modificar un dato borrando su versión anterior y agregando la nueva.

Veamos cómo podemos borrar datos. Supongamos que nos hemos cansado de la acepción de la palabra "casa" y la deseamos borrar. El problema que tienen los archivos secuenciales es que para borrar un dato almacenado "en el medio" tenemos que copiar el archivo íntegro.

Vamos a utilizar el método "padre-hijo" para borrar los datos. El esquema de trabajo será:

1) Fijar en modo lectura el archivo principal o "maestro".
2) Crear un archivo "maestro actualizado" en donde se copiarán los datos que no se borrarán (abrirlo y fijarlo para escritura).
3) Leer el archivo "maestro". Si el dato leído no coincide con el dato a borrar, se copia en el maestro actualizado. Caso contrario, se lee el siguiente dato.
4) Una vez copiado el "maestro actualizado", se copia el archivo temporario al "maestro" definitivo, borrándose luego el archivo temporario.

Veamos cómo se implementa:

```
para borra :borrada
cerrartodo
esc lista "Borrando: :borrada
abrir :nombrear
fleer :nombrear
abrir "copia
fesc "copia
borrando :borrada
cerrartodo
boarchivo :nombrear
esc [Copiando nuevo archivo...]
abrir "copia
fleer "copia
abrir :nombrear
fesc :nombrear
copiarnuevo
esc [Cumplido]
inicializar :nombrear
```

```
fin

para borrando :que
si leerfda? [bon "linea parar]
hacer "línea 11.a
si no igual? ( primero :linea ) :que [esc.a :linea esc
( lista :linea [- se copia]]
borrando :que
fin
```

```
para copiarnuevo ,
si leerfda? [cerrartodo boarchivo "copia parar]
esc.a 11.a
copiarnuevo
fin
```

Como subproducto de este método, se notará que si se indica una palabra que no existe en el diccionario, lo único que se logra es una copia del diccionario - nada indica el error.

La forma de utilizar este comando es:

```
?borra "palabra
```

Borremos la palabra "casa":

```
?borra "casa
Borrando: casa

(nos muestra la lista de lo que se copia)

Copiando nuevo archivo...
Cumplido
?
```

Como toda historia con final feliz, *dicologo* tiene su despedida: el procedimiento adios. Veamos de qué se trata:

```
para adios
cerrartodo
fin
```

Sencillamente, lo que hacemos es cerrar todos los archivos abiertos en este momento. También se puede cerrar un archivo en particular. Podríamos cambiar "cerrartodo" por el siguiente comando:

```
cerrar :nombrear
```

Con esto finalizamos el desarrollo de este entorno. Nótese que nuestro *dicologo* permite generar cualquier tipo de base de datos.

Presentamos el listado completo del DICOLOGO en el Apéndice A.

Copia de pantallas por impresora: El PRELOGO como micromundo.

Vamos a tratar ahora la extensión de copia de pantallas por impresora. Esta extensión permite, impresora gráfica mediante, obtener los diseños y dibujos que tan trabajosamente se hayan desarrollado en el entorno LOGO sin necesidad de utilizar papel de calcar y/o métodos análogos.

Crearemos un micromundo que nos permitirá darle un uso interesante a esta posibilidad y que, a su vez, extenderá el uso de la computadora.

Es obvio que el LOGO necesita por lo menos que el usuario conozca las letras y los números; lo que deja sin posibilidades a los pequeñitos. Para remediar este problema haremos un PRELOGO que les permitirá dibujar sus creaciones conduciendo a la tortuga con sólo pulsar algunas teclas. Para facilitar las cosas podríamos pegar sobre las teclas que se usen etiquetas con símbolos como flechas para las direcciones, o el dibujo de un lápiz y un lápiz tachado para las instrucciones CP y SP, O una goma de borrar y una señal de PARE para terminar de dibujar y poder escribir entonces un nombre para el dibujo (evidentemente, una persona mayor se encargará de escribir lo que el niño desee).

Este PRELOGO es de gran utilidad para los maestros de preescolar hasta3er. grado que quieren iniciar a sus alumnos en el uso de la computadora.

Iremos describiendo los diversos procedimientos que integran el PRELOGO, suponiendo un conocimiento mínimo sobre MSX-LOGO.

Lecturas del teclado.

En esta programa introducimos el concepto de leer los datos que se introducen desde el teclado en el curso de las instrucciones que se realizan en un procedimiento. Este es el caso de **1c** y **11**.

La primera primitiva lee un carácter (leercaracter) que, como vemos en el siguiente procedimiento lo guarda en la variable de nombre **"t**, para luego comparar el contenido de esta variable con diferentes letras que corresponden a teclas donde podemos tener nuestras etiquetas pegadas; y realiza las instrucciones adecuadas en el caso de que lo que hay en **t** sea igual a alguna de esas letras.

Veamos el procedimiento:

```
para teclas
hacer "t 1c
si :t = "a [ad 10 hacer "instrucciones ponult [ad10]
:instrucciones]
si-:t = "r [at 10 hacer "instrucciones ponult [at10]
:instrucciones]
si :t = "d [de 45 hacer "instrucciones ponult [de45]
:instrucciones]
si :t = "i [iz 45 hacer "instrucciones ponult [iz45]
:instrucciones]
si :t = "l [cpfcolor 15 hacer "instrucciones
ponult[cpfcolor 15] :instrucciones]
si :t = "n [spfcolor 1 hacer "instrucciones ponult
[spfcolor 1] :instrucciones]
si :t = "g [hacer "instrucciones menosultimo
instrucciones bp repite :instrucciones]
si :t = "s [nombra]
si :t = "c [fcolor 3 copiarpantalla fcolor 15]
si :t = "f [bp esc [Salimos del PRELOGO] parar]
teclas
fin
```

En el procedimiento nombra utilizamos la primitiva que lee una lista(leerlista) y la primera palabra de esa lista la guardamos en la variable **"nombre** para utilizarla luego en la definición del procedimiento. El procedimiento es el que sigue:

```
para nombra
```

```
escribir [QUE NOMBRE LE QUIERES PONER AL DIBUJO?]
hacer "nombre primero 11
si definido? :nombre [escribir frase :nombre [YA ESTA
DEFINIDO!] esperar 100 bp repite :instrucciones parar]
definir palabra y «nombre ponpri [] instrucciones
escribir frase [ACABAS DE DEFINIR] :nombre
esperar 150
bp
fin
```

La mayoría de los LOGOS sólo tienen estas dos primitivas **1c** y **11**perosi quisiéramos construir la primitiva 1p (leerpalabra) podríamos hacer el siguiente procedimiento:

```
para leerpalabra
resp primero 11
fin
```

ó bien:

```
para lp
  resp primero 11
fin
```

Hemos creado una primitiva que es una operación que devuelve el resultado de la lectura de lo que se ha escrito desde el teclado. En nuestro ejemplo este resultado es el contenido que le daremos a la variable.

Condicionales.

En el procedimiento tecla decíamos que el resultado de pulsar una tecla lo guardábamos en una variable para compararlo con diferentes letras. Esta comparación la realizamos con diferentes condicionales; es decir:

si ocurre tal situación entonces hacer tal cosa, sinó continuar con la siguiente instrucción.

La sintaxis adecuada para realizar esto es la siguiente:

si condición (lista instrucciones 1] [lista instrucciones 2]

Cuando la condición es cierta se ejecuta la primera lista de instrucciones y si es falsa la segunda. La ausencia de la segunda lista de instrucciones no afecta al condicional y en el procedimiento simplemente continuará sin haber realizado la primera lista de instrucciones.

Una condición puede ser de diferente tipo. Por ejemplo, en teclas es una igualdad. El contenido de t igual a una letra.

Pero también podemos tener condiciones de otro tipo como algo menor o mayor (< ó >) que otra cosa **(:x > 10 :Y < :X)** o utilizar otras primitivas LOGO a las que se les llama predicados y que son fáciles de reconocer porque por lo general llevan un signo de interrogación al final de la palabra, como es el caso de **vacio?** en el procedimiento **repite**. Estos predicados son operaciones que devuelven los valores **cierto** o **falso** y que son la primera entrada de datos para el comando **si**. Podemos probar por ejemplo con:

lista? "casa

y obtendremos un mensaje:

`No sé qué hacer con falso`, o
`palabra? "gato` y será
`No sé qué hacer con cierto`

También podemos tener condiciones más complejas utilizando los llamados operadores lógicos como **`y`**, **`o`** y **`no`**.

Su sintaxis es la siguiente:

`si y :X > :y :x < :z [suma :x :z]` ó
`si no vacio? :nombre [repetir 4 [ad 10 de 90]]`

Estas son operaciones que en el caso de **`y`** y**`o`** necesitan de dos entradas y no cambia el valor de verdad del predicado negándolo.

El uso que en el procedimiento **`repite`** hacemos del condicionales muy importante, porque es la forma en que detenemos la llamada *recursiva*, es decir, cuando volvemos a empezar el procedimiento en la última línea y le vamos quitando un elemento al dato de entrada con **`menosprimero`**. Hay un momento en que estará vacío y es allí cuando la instrucción que utilizamos es **`parar`**.

Ejecución y definición de procedimientos dentro de un procedimiento.

Ya sabemos que para ejecutar un procedimiento basta escribirlo y pulsar<RETURN> para que se realice; y para definirlo usamos **para** y **fin**.

Pero es que también podemos ejecutar procedimientos y definir procedimientos dentro de los procedimientos que estamos definiendo. Aquí en el PRELOGO lo utilizamos en **repite**.

La primitiva cumplir necesita como entrada una lista y el contenido de nuestra variable **instrucciones** no es otra cosa que una serie de listas que hemos ido guardando en **teclas**.

[[ad 10] ad 10] ad 10] [de 45]...[at 10] [cp]]

Esto lo podremos ver luego de hacer algún dibujo escribiendo:

mostrar :instrucciones

Por eso es que para borrar sacamos de **instrucciones** la última instrucción guardada en forma de lista y llamamos a **repite** luego de borrar la pantalla y se van ejecutando lista por lista con **cumplir primero :datos.**

Esta forma de borrar, digámosle histórica, tiene importancia para que el niño vea el proceso de creación de su dibujo y las quizás innecesarias acciones que ha realizado.

El procedimiento **nombra** es en donde utilizamos la otra técnica para definir procedimientos.

Para ello nos sirve la primitiva definir que es un comando con dos entradas: la primera es un nombre y la segunda una lista.

En nuestro procedimiento el nombre lo introducimos desde el teclado y la sintaxis exige que éste tenga comillas por delante.

```
definir "nombre [[] [] []]
```

Por eso hacemos **palabra " :nombre** ya que palabra une las comillas al contenido de **nombre**.

En cuanto a la segunda entrada que debe ser una lista se nos presenta la situación que los procedimientos pueden tener datos o no y por ello la lista presentará las siguientes características:

Para definir un procedimiento mediante definir y que no tiene ningún dato variable como entrada deberá ser una lista cuyo primer elemento sea una lista vacía.

Por ejemplo, el siguiente procedimiento:

```
para circulo
  repetir 360 [ad 1 de 1]
fin
```

se puede definir así:

```
definir "circulo [ [] [repetir 360 [ad 1 de 1]]]
```

Pero si debemos definir un procedimiento con datos variables los datos serán listas cuyo contenido será el nombre de los datos.

```
para estrella :lado
  repetir 5 [ad :lado de 144]
fin
```

se define en este caso:

```
definir "estrella [[lado] [repetir 5 [ad :lado de 144]]]
```

En nuestro procedimiento el truco es poner en primer lugar de la variable instrucciones una lista vacía:

```
definir palabra " :nombre pp [] :instrucciones
```

Esta posibilidad de definir "automáticamente" procedimientos en el PRELOGO puede permitir ir enseñando a los niños los nombres de los objetos que previamente dibuja por lo que es una herramienta de introducción a la escritura muy interesante en las primeras edades.

Este entorno básico puede contemplarse con la introducción de nuevas funciones que en el trabajo con los niños hemos notado de importancia.

Por ejemplo, la función de copiar pantalla nos permite obtener el dibujo que ha creado el niño en un papel (de impresora), con lo cual siempre será posible recortar y luego armar una colección de figuritas creadas por él mismo.

Para utilizar este entorno, bastará con ingresar **`prelogo`** una vez cargado el programa.

Las teclas que se utilizan son:

`<a>:`*avanzar*
`<r>:`*retroceder*
`<d>:`*derecha*
`<i>:`*izquierda*
`<l>:`*equivale a conpluma (cp)*
`<n>:`*equivale a sinpluma (sp)*
`<g>:`*elimina la última instrucción agregada*
`<s>:`*permite nombrar un dibujo (creando además el procedimiento)*
`<c>:`*copiar pantalla por impresora*
`<f>:`*salir del PRELOGO.*

Como sugerencia de desarrollo, podría tratar de agregar a los comandos el manejo de palanca de mando (joystick) y la posibilidad de agregar procedimientos ya definidos dentro del procedimiento que se va creando en **`:instrucciones`**.

Listado completo del PRELOGO:

Para observar todos los procedimientos descriptos, véase el Apéndice A.

Sistema Musical MSX LOGO

El sistema básico MSX-LOGO permite escribir y ejecutar todo tipo de melodías, usando hasta tres voces simultáneamente, mediante la primitiva **sonido**.

Para facilitar la escritura de obras complejas, se ha creado un sistema musical con nuevas órdenes, **iniciar.música** y **música**, que permiten escribir melodías con los signos musicales usuales, evitando la necesidad de consultar una tabla de frecuencias o hacer cálculos matemáticos para determinar la duración de las notas.

La orden música acepta como argumentos los nombres de las notas de la escala cromática (do, do#, re, etc.) y las abreviaturas de las figuras rítmicas conocidas (r=redonda, b=blanca; n=negra; etc.), y además permite definir el volumen y la velocidad de un trozo musical completo, sin necesidad de hacerlo para cada nota.

El sistema transforma luego las notas así escritas con la orden música en las órdenes **sonido** del sistema básico.

Vamos a describir cómo se utiliza el Sistema Musical MSX-LOGO.

Descripción.

El sistema musical MSX-LOGO ha sido creado para facilitar la escritura de obras musicales. Consta de tres partes: una de inicialización, otra de creación y una tercera de ejecución de la obra.

Inicialización:

Para comenzar a escribir una obra musical, lo primero que se debe hacer es **recordar** el archivo **MUSICA** que contiene todos los procedimientos.

Luego se ejecutará el procedimiento **iniciar.música** que define las frecuencias de las notas que se van a utilizar. Sólo se define la octava central, ya que las restantes octavas se calculan automáticamente.

Una vez ejecutado este procedimiento, ya está todo listo para comenzar a escribir la obra musical.

Creación:

Para definir las notas de una melodía utilizamos el procedimiento **música**. Este procedimiento convierte las notas que se ingresan comodato en las órdenes **sonido** conocidas.

Por ejemplo:

```
música "partel 150 10 [[sol c. 0] [sol sc 0] [la n 0] [sol n 0] [do 5 n 0] [si b 0] ]
```

Vemos que la orden **música** tiene varios argumentos que describiremos a continuación.

Nombre de la melodía:

El primer argumento es el nombre que identifica a la melodía que vamos a definir. Puede ser un trozo de una canción o una melodía completa. Para hacer más sencilla la escritura recomendamos que se escriban frases cortas dentro de cada orden **música**.

En el ejemplo, **"partel** es el nombre de un trozo de una melodía que luego ejecutaremos dentro de un procedimiento.

Velocidad:

El segundo argumento es la velocidad de ejecución. Modificando este argumento se puede variar el "tempo" de una obra musical o de un trozo

de ella. Este valor es válido para todas las notas que se especifiquen dentro de la orden música.

Si le asignamos una velocidad 50, la duración real de una redonda será de un segundo. O sea, una blanca durará 1/2 segundo, una negra 1/4 de segundo, etc. Cuanto menor sea el número que le asignamos, más rápida será la ejecución. Una velocidad 100 hará durar 2 segundos a una redonda, 1 segundo a una blanca, 1/2 a una negra, etc.

Volumen:

Es el tercer argumento e incide en todas las notas especificadas dentro de la orden música. Su rango varía entre 0 y 15. Su funcionamiento es similar al que tiene en la orden sonido. En caso de escribir un valor mayor que 15 el sistema emite el mensaje:

el volumen es superior al máximo permitido

En ese caso (así como en todos los casos en que aparece algún error), se debe corregir y volver a ejecutar.

Lista de notas:

El cuarto argumento es una lista compuesta con listas de todas las notas que forman parte del trozo musical que se desea definir. Cada sublista contiene cuatro elementos, que son: el nombre de la nota, el número de octava a la que pertenece, el nombre de la figura rítmica (duración de la nota) y el número de canal por donde se va a emitir.

Nombre de la nota:

Las notas permitidas son las de la escala cromática y son las siguientes:

do do# reb re re# mib mi fa fa# solb sol sol# lab la la# sib si y sil (silencio).

En caso de necesitar una nota con doble sostenido (++) o con doble bemol (bb) se debe reemplazar por su equivalente en la escala. Por ejemplo, en lugar de fa##, escribir sol.

Si se quiere escribir un silencio se utiliza sil que emitirá una nota de la duración especificada, pero el sistema, automáticamente, le asigna un volumen 0.

Si se escribe mal el nombre de una nota, el sistema emite el mensaje:

no existe ninguna nota llamada ...

Número de octava:

A continuación del nombre de la nota se debe escribir el número de octava a la que pertenece. Son permitidas hasta 7 octavas que van desde el do con frecuencia = 33 hasta el si con frecuencia = 3952. La octava central es la número 4. Si las notas a definir pertenecen a la octava central (equivalente a la octava central del piano), no hace falta especificar el número de octava. El sistema, a partir del nombre de la nota y del número de octava, calculará automáticamente la frecuencia de la nota a emitir.

Cuando se utiliza por error un número de octava superior a 7, el sistema emite el mensaje:

`la octava elegida es superior a la máxima permitida en ...`

Nombre de la figura rítmica:

El tercer elemento de la sublista (o segundo en caso de estar en la octava central) es el nombre de la figura rítmica que corresponde con una nota en el orden en que fueron escritas.

Los nombres de las figuras permitidas y sus valores asociados son:

redonda	**r**	**(1)**
blanca con puntillo	**b.**	**(1,33)**
blanca	**b**	**(2)**
negra con puntillo	**n.**	**(2,66)**
negra	**n**	**(4)**
corchea con puntillo	**c.**	**(5,33)**
corchea	**c**	**(8)**
semicorchea con puntillo	**sc.**	**(10,66)**
semicorchea	**sc**	**(16)**
fusa con puntillo	**f.**	**(21,33)**
fusa	**f**	**(32)**
semifusa	**sf**	**(64)**
tresillo de redondas	**tr**	**(1,5)**

tresillo de blancas	**tb**	**(3)**
tresillo de negras	**tn**	**(6)**
tresillo de corcheas	**tc**	**(12)**
tresillo de semicorcheas	**tsc**	**(24)**
tresillo de fusas	**tf**	**(48)**
tresillo de semifusas	**tsf**	**(96)**

Las figuras tresillo de redonda, de blanca, etc. se han agregado para permitir escribir tresillos en compases simples. Por ejemplo, **tf** equivale a una de las fusas que componen un tresillo de fusas.

En caso de escribir mal el nombre de la figura rítmica, el sistema emite el mensaje:

no existe ninguna figura rítmica llamada ...

La duración real de cada nota dependerá del valor asignado al parámetro velocidad y a la duración determinada por la figura rítmica.

Número de canal:

El último elemento de la sublista es el número de canal por donde se desea emitir la nota especificada.

Como sabemos, el sistema MSX-LOGO permite emitir sonidos por tres canales simultáneamente. Ellos son los canales 0, 1 y 2.

Por ejemplo, si deseamos transcribir una partitura a tres voces, debemos hacer primero una lectura vertical de la partitura y luego una horizontal, es decir, iremos describiendo los acordes que se forman, emitiendo cada nota del acorde por un canal diferente. En caso de que las notas tengan duraciones diferentes, se escribirán en primer lugar las notas que se deben articular primero.

Si el número de canal no es 0, 1 o 2, el sistema emitirá el siguiente mensaje:

código de canal no permitido en ...

El sistema verifica además que cada sublista contenga por lo menos tres elementos (o sea, nota, figura rítmica y número de canal), y en caso contrario emite el mensaje:

falta algún argumento en la lista ...

Ejecución:

Cuando se termina de ejecutar la orden música, aparece un mensaje que indica que el procedimiento determinado por el nombre de la melodía ya está definido.

Con solo escribir el nombre de dicho procedimiento se podrá escuchar la obra musical creada.

Si hemos dividido la obra en varias partes, cada una creada por la orden música, podemos escribir ahora un procedimiento que nombre a cada una de las partes y al ejecutarlo se oirá una obra completa.

Veamos un ejemplo:

Primero recordamos el archivo **MUSICA** y ejecutamos **iniciar.música**.

Luego escribimos el procedimiento:

```
para crear.cumple
música "parte1 150 10 [[sol 5,33 0] [sol 16 0][la 4 0] [sol 4 0] [do 5 4 0] [si 2 0]]
música "parte2 150 10 [[sol 5,33 0] [SOL 16 0]][la 4 0] [sol 4 0] [re 5 4 0] [do 5 2 0]]
música "parte3 150 10 [[sol 5,33 0] [sol 16 0][sol 5 4 0] [mi 5 4 0] [do 5 8 0] [do 5 8 0] [si4 0] [la 20]]
música "parte4 150 10 [[fa 5 5,33 0] [fa 5 16 0](mi 5 4 0] [do 5 4 0] [re 5 4 0] [do 5 2 0]]
fin
```

y lo ejecutamos:

```
?crear.cumple
```

Aparecen los mensajes:

```
parte1 está definido
parte2 está definido
parte3 está definido
parte4 está definido
```

Podemos escribir ahora el procedimiento:

```
para feliz.cumple
parte1
parte2
parte3
parte4
fin
```

Si ordenamos **feliz.cumple** escucharemos la canción que hemos creado.

Veamos ahora otro ejemplo, más elaborado, donde se utilizan dos voces. Es un trozo de un aria para soprano y barítono de La Flauta Mágica de Mozart.

```
para crear.Mozart
crear.Mozart1
crear.Mozart2
crear.Mozart3
crear.Mozart4
fin
```

```
para crear.Mozart1
música "parte1A 192 10 [[sil n 0] [sib c 0] [sib c 0] [sol c 0] [sol c 0] [sol c 0] [mib c 0] [mib c 0] [re c 0] [fa c 0] (lab c 0] [lab c 0] [sol c 0]]
música "parte2A 192 10 [[sib c 0] [sib c 0] [mib 5 c 0] [re 5c 0] [re 5c 0] [do 5 c 0] [sib c 0] [sib c 0] [lab c 0] [sol c 0] [fa n 0]]
música "parte3A 192 10 [[sib 3 c 0] [sib 3 c 0] [sol 3 c 0] [sol 3 c 0] [sol 3 c 0] [mib 3 c 0] [mib 3 c 0] [re 3 c 0] [fa 3 c 0] [lab 3 C 0] [lab 3 c 0] [sol 3 c 0]]
```

```
música "parte4A 192 10 [[sib 3 c 0] [mib c 0] [sil c 0] [re c 0] [do c 0] [re sc 0] [sib 3 c0] [sib 3 c. 0] [do sc 0] [la 3 c 0] [sib 3 c 0][sil c 0]]
fin

para crear.Mozart2
música "parte5A 192 9 [[sib c 0] [sib 3 c 1] [fa5c 0] [lab3 n 1] [re 5 c 0] [sib c 0] [sib 3 c1] [mib 5 n 0] [sol 3 n 1]]
música "parte6A 192 8 [[mib 5 c 0] [sib 3.Cc 1][fa 5 c 0] [lab 3 n 1] [re 5 c 0] [sib c 0] [sib3 c 1] [mib 5 n 0] [sol 3 n 1]]
fin

para crear.Mozart3
música "parte7A 192 7 [[sil c 0] [sil c 1] [mib 5c 0] [do c 1] [mib 5 c 0] [do c 1] [mib 5 c 0][do c 1] [fa 5 c 0] [lab 3 e 1] [sil c 0] [sil c1]]
música "parte8A 192 7 [[fa 5 c O] [lab 3 c 1] [sib c. 0] [sib 3 c 1] [sil c 1] [do 5 sc 0] [re5 c 0] [sib 3 c 1] [mib 5 c 0] [do c 1][sil n O] [sil n 1]]
fin

para crear.Mozart4
música "parte9A 192 9 [[mib 5 c 0] [sol 3 c 1][re 5 sc 0] [sol 3 c 1] [mib 5 sc 0] [re 5 sc 0][sol 3 c 1] [mib 5 sc 0] [fa 5 n 0] [lab 3 n 1]]
música "partel0A 192 9 [[lab 5 sc 0] [lab 3 c 1][fa 5 sc 0] [mib 5 n 0] [sib 3 n 1] [fa 5 sc 0][sib 3 c 1] [re 5 sc 0] [mib 5 n 0] [mib 3 n 1]]
fin
```

Si ejecutamos **crear.Mozart**, el sistema creará los siguientes procedimientos:

```
para parte1A
sonido 0 28 0 48
sonid0 0 466 10 24
sonido 0 466 10 24
sonido 0 392 10 24
sonido 0 392 10 24
sonido 0 392 10 24
sonido 0 311 10 24
sonido 0 311 10 24
sonido 0 294 10 24
sonido 0 349 10 24
sonido 0 415 10 24
sonido 0 415 10 24
sonido 0 392 10 24
fin

para parte2A
sonido 0 466 10 24
sonido 0 466 10 24
sonido 0 622 10 24
sonido 0 588 10 24
sonido 0 588 10 24
sonido 0 524 10 24
sonido 0 466 10 24
sonido 0 466 10 24
sonido 0 415 10 24
sonido 0 392 10 24
sonido 0 349 10 48
fin

para parte3A
sonido 0 233 10 24
sonido 0 233 10 24
sonido 0 196 10 24
sonido 0 196 10 24
sonido 0 196 10 24
sonido 0 155 10 24
sonido 0 155 10 24
sonido 0 147 10 24
sonido 0 174 10 24
sonido 0 207 10 24
sonido 0 207 10 24
sonido 0 196 10 24
fin

para parte4A
sonido 0 233 10 24
sonido 0 311 10 24
sonido 0 28 0 24
sonido 0 294 10 24
sonido 0 262 10 36
sonido 0 294 10 12
sonido 0 233 10 24
sonido 0 233 10 36
sonido 0 262 10 12
sonido 0 220 10 24
sonido 0 233 10 24
sonido 0 28 0 24
fin

para parte5A
sonido 0 466 9 24
sonido 1 233 9 24
sonido 0 698 9 24
sonido 1 207 9 48
sonido 0 588 9 24
sonido 0 466 9 24
sonido 1 233 9 24
sonido 0 622 9 48
sonido 1 196 9 48
fin

para parte6A
sonido 0 622 8 24
sonido 1 233 8 24
sonido 0 698 8 24
sonido 1 207 8 48
sonido 0 588 8 24
sonido 0 466 8 24
```

```
sonido 1 233 8 24
sonido 0 622 8 48
sonido 1 196 8 48
fin
para parte7A
sonido 0 28 0 24
sonido 1 28 0 24
sonido0 622 7 24
sonido 1 262 7 24
sonido0 622 7 24
sonido 1 262 7 24
sonido0 622 7 24
sonido 1 262 7 24
sonido 0 698 7 24
sonido 1 207 7 24
sonido 0 28 0 24
sonido 1 28 0 24
fin

para parte8A
sonido 0 698 7 24
sonido 1 207 7 24
sonido 0 466 7 36
sonido 1 233 7 24
sonido 1 28 0 24
sonido 0 524 7 12
sonido0 588 7 24
sonido 1 233 7 24
sonido0 622 7 24
sonido 1 262 7 24
sonido0 28 0 48
sonido 1 28 0 48
fin

para parte9A
sonido0 622 9 24
sonido 1 196 9 24
sonido 0 588 9 12
sonido 1 196 9 24
sonido 0 622 9 12
sonido 0 588 9 12
sonido 1 196 9 24
sonido 0 622 9 12
sonido 0 698 9 48
sonido 1 207 9 48
fin
para parte10A
sonido0 830 9 12
sonido 1 207 9 24
sonido 0 698 9 12
sonido0 622 9 48
sonido 1 233 9 48
sonido0 698 9 12
sonido 1 233 9 24
sonido0 588 9 12
sonido0 622 9 48
sonido 1 155 9 48
fin
```

Ahora podemos crear el procedimiento Mozart, que ejecute las partes ya definidas:

```
para Mozart
parte1A
parte2A
parte3A
parte4A
parte5A
parte6A
parte7A
parte8A
parte9A
parte10A
fin
```

Para oir lo que hemos creado sólo hace falta escribir **Mozart**.

Para poder guardar las melodías que hemos definido, el sistema contiene una orden, **desocupar**, que borra todos los procedimientos del sistema musical que están en memoria. Así evitamos guardar cada vez los procedimientos del sistema y sólo guardamos los que hayamos definido.

Por ejemplo, si deseamos guardar el procedimiento **Mozart**, escribimos:

```
?desocupar
?guardar "MOZART
```

y creamos un archivo llamado **MOZART** que contiene los procedimientos que hemos definido.

Para volver a escuchar nuestra obra, sólo hace falta escribir:

```
?recordar "mozart
?Mozart ;
```

Para analizar los procedimientos del sistema musical MSX-LOGO, verlos listados del Apéndice A.

El LOGO en el espacio de tres dimensiones: LOGO TRidimensional

La última extensión que incluimos en este paquete es el LOGO TRI, que permite que el MSX-LOGO maneje coordenadas espaciales en tres dimensiones. Dado que el LOGO TRI se explica intensivamente en el manual que acompaña al TALENT MSX-LOGO, sólo incluimos una breve referencia de los conceptos que involucran el manejo del espacio de tres dimensiones en MSX-LOGO.

Con la extensión tri del MSX-LOGO se pueden crear gráficos tridimensionales, con el objeto de dibujar figuras en perspectiva sobre la pantalla. Esto nos permite crear un micromundo lleno de posibilidades, desde la visualización de objetos tridimensionales, estudios de perspectiva, desarrollo del sentido de orientación espacial, creación de objetos en el espacio, etc.

Para poder realizar figuras en el espacio hay que dotar a la tortuga de LOGO de procedimientos que posibiliten su colocación y orientación en el espacio tridimensional, y, por tanto, necesitamos como referencia unos ejes de coordenadas espaciales que llamaremos tríada.

Colocar la tortuga en un punto dado por sus tres coordenadas no es más que un problema simple de proyección sobre el plano de la pantalla de la computadora (que supondremos el plano X Y).

Para poder dotarla de la posibilidad de giros en el espacio es preciso definir los tres giros básicos en un sistema de tres ejes, uno alrededor de cada eje, que se denomina, según la denominación utilizada por la técnica aeroespacial: rolar, virar y cabecear.

A partir de este momento, la tortuga tridimensional es un plano orientado, más que una semirrecta orientada en el plano. Sin embargo, deberemos definir la perspectiva desde la cual podremos "ver" a nuestra tortuga espacial.

El detalle de la convención utilizada y la ejemplificación se encuentran en el manual del Talent MSX-LOGO a partir del Capítulo 6.

En el Anexo D del mismo manual se encuentra un listado completo de los procedimientos que componen al LOGO TRI, que también se detallan en el Apéndice B de este manual.

Apéndice A - Listados.

En este apéndice se incluyen todos los listados de los procedimientos que conforman las distintas extensiones que han sido desarrolladas en MSX-LOGO.

Listado No. 1 - DICOLOGO (para extensiones de manejo de archivos).

```
para adios
cerrartodo
fin

para agrega
esc [quieres agregar algo mas? si - no]
si 11 = (no] [parar]
esc [escribe la palabra y su relacion)
escribir.a 11
agrega
fin

para borra :borrada
cerrartodo
esc lista "Borrando: :borrada
abrir :nombrear
fleer :nombrear
abrir "copia
fesc "copia
borrando :borrada
cerrartodo
boarchivo :nombrear
esc [Copiando nuevo archivo...]
abrir "copia
fleer "copia
abrir :nombrear
fesc :nombrear
copiarnuevo
esc [Cumplido]
inicializar :nombrear
fin
```

```
para borrando :que
si leerfda? [bon "linea parar]
hacer "linea 11.a
si no igual? ( primero :linea ) :que [esc.a :linea esc
( lista :linea [- se copia]]
borrando :que
fin

para buscar
esc [que palabra quieres buscar?]
hacer "palabra primero 11
fposleer 0
busqueda
fin

para busqueda
si leerfda? [parar] [hacer "dic 11.a]
si :palabra = primero :dic [esc :dic parar]
busqueda
fin

para copiarnuevo
si leerfda? [cerrartodo boarchivo "copia parar]
esc.a 11.a
copiarnuevo
fin
```

```
para inicia
esc [Nombre archivo a utilizar?]
hacer "nombrear primero 11
si vacia? :nombrear [parar] [inicializar :nombrear]
fin

para inicializar :nombrear
abrir :nombrear
fleer :nombrear
fesc :nombrear
fin
```

```
para listar
si leerfda? [parar]
hacer "ver 11.a
si no vacia? :ver [esc :ver]
listar
fin

para muestra.todo
fposleer 0
listar
fin
```

Listado No. 2- PRELOGO (para extensiones de copia de pantalla por impresora).

```
para nombra
escribir [QUE NOMBRE LE QUIERES PONER AL DIBUJO?]
hacer "nombre primero 11
si definido? :nombre [escribir frase :nombre [YAESTA
DEFINIDO!] esperar 100 bp repite :instrucciones parar]
definir palabra y " :nombre ponpri [] :instrucciones
escribir frase [ACABAS DE DEFINIR] :nombre
esperar 150
bp
fin

para prelogo
bp mt cp
hacer "instrucciones []
teclas
fin

para repite :datos
si vacia? :datos [parar]
cumplir primero :datos
repite menosprimero :datos
fin
```

```
para teclas
hacer "t lc
si :t = "a [ad 10 hacer "instrucciones ponult [ad10]
:instrucciones]
si :t = "r [at 10 hacer "instrucciones ponult [at10]
:instrucciones]
si :t = "d [de 45 hacer "instrucciones ponult [de45]
:instrucciones])
si :t = "i [iz 45 hacer "instrucciones ponult [iz45] :
instrucciones]
si :t = "l [cpfcolor 15 hacer "instrucciones ponult
[cpfcolor 15] :instrucciones]
si :t = "n [spfcolor 1 hacer "instrucciones ponult
[spfcolor 1] instrucciones]
si :t = "g [hacer "instrucciones menosultimo
"instrucciones bp repite : instrucciones]
si :t = "s [nombra]
si :t = "c [fcolor 3 copiarpantalla fcolor 15]
si :t = "f [bp esc [Salimos del PRELOGO] parar]
teclas
fin
```

Listado No. 3 - Procedimientos del Sistema Musical MSX-LOGO.

```
para armar.nota
hacer "frecuencia cosa :altura
si :octava > 4 [repetir ( :octava - 4 ) [hacer
"frecuencia :frecuencia * 2]]
si :octava < 4 [repetir ( 4 - :octava ) [hacer
"frecuencia ent ( :frecuencia / 2 )]]
hacer "duración ent ( :tempo / cosa «ritmo )
si :altura = "sil (hacer :canción ponúlt ( frase
"sonido :canal :frecuencia 0 «duración ) cosa :canción
parar]
hacer :canción ponúlt ( frase "sonido ¿canal
:frecuencia :volumen :duración ) cosa :canción
fin

para asignar :11 :12
si vacía? :11 [parar]
hacer primero :11 primero :12
```

```
asignar mp :11 mp :12
fin

para control :n :l
si :l = [] [resp "falso]
si :n = primero :1 [resp "cierto]
resp control :n ( mp :1 )
fin

para desglosar
si ( cuenta :nota ) < 3 [( esc [falta algún argumento
en la lista] :nota ) nivelsuperior]
hacer "altura primero :nota
hacer "canal último :nota
hacer "ritmo último mu :nota
si ( cuenta :nota ) = 3 [hacer "octava 4 parar]
hacer "octava item 2 :nota
fin

para desocupar
bons
bo   ([iniciar.música   asignar   música   interpretar
desglosar verificar control armar.nota desocupar]
fin

para iniciar.música
hacer "lnotas [do do# reb re re# mib mi fa fa# solb
sol sol# lab la la# sib si sil]
asignar :lnotas [262 277 277 294 311 311 330 349370
370 392 415 415 440 466 466 494 28]
hacer "lritmos [r bn c sc f sf b. n. c. sc. f. tr tb
tn tc tsc tf tsf]
asignar :lritmos [1 2 4 8 16 32 64 1,33 2,66 5,33
10,66 21,33 1,5 3 6 12 24 48 96]
fin
```

```
para interpretar :ln
si :ln = [] [parar]
hacer "nota primero :ln
desglosar
verificar
armar.nota
interpretar mp :ln
fin

para música :canción :tempo :volumen :1n
hacer :canción []
interpretar :ln
definir :canción ponpri [] cosa :canción
( esc :canción [está definido] )
bon :canción
fin

para verificar
si ( control :altura :lnotas ) = "falso [( esc[no
existe ninguna nota llamada] :altura ) nivelsuperior]
si :octava > 7 [( esc [la octava elegida es superior a
la máxima permitida en] :nota ) nivelsuperior]
si ( control :ritmo :lritmos ) = "falso [( esc[no
existe ninguna figura rítmica llamada] :ritmo)
nivelsuperior]
si :canal > 2 [( esc [código de canal no permitido en]
:nota ) nivelsuperior]
si :volumen > 15 [esc [el volumen es superior al
máximo permitido] nivelsuperior]
fin
```

Apéndice B
Primitivas y procedimientos que incorporan las extensiones al Talent MSX LOGO.

A continuación se ofrece un listado ordenado alfabéticamente de todas las primitivas que se implementan a través de las extensiones al Talent MSX LOGO.

El formato utilizado para la descripción de las primitivas es el siguiente:

primitiva (argumento) (tipo, entorno, archivo)

Donde tipo indica si es comando u operación, entorno indica en qué entorno funciona la primitiva y archivo indica algún comentario adicional sobre la misma. El argumento muestra el tipo de argumento que requiere la primitiva, si es que lo tiene. Cuando se indica objeto se quiere representar a cualquier objeto logo (palabras, listas, etc.).

---

aad (número) (comando, logotri, breves)
Significa andar adelante. Equivale a **andar**.

---

aat (número) (comando, logotri, breves)
Significa andar atrás. Equivale a **andar** con argumento negativo.

---

abrir (archivo) (comando, archivos, primit.oem)
Prepara el archivo indicado por su argumento para la lectura o escritura de datos. En general, será la primera orden que haga referencia al archivo.

Ejemplo:
```
?abrir "diccio
```

andar (número) (comando, logotri, tri)
Mueve la tortuga tridimensional en la dirección longitudinal la distancia indicada por su argumento, sin que varíe su plano ni su orientación, al igual que en el caso de la tortuga plana clásica.

Ejemplo:
?andar 50

---

cabecear (número) (comando, logotri, tri)
Gira la tortuga tridimensional según su eje transversal el ángulo indicado por su argumento.

Ejemplo:
?cabecear 60

---

cad (número) (comando, logotri, breves)
Significa cabecear adelante. Equivale a **cabecear**.

---

cat (número) (comando, logotri, breves)
Significa cabecear atrás. Equivale a **cabecear** con argumento negativo.

---

cerrar (archivo) (comando, archivos, primit.oem)
Cierra el archivo especificado por su argumento **cerrar** concluye previamente cualquier operación de lectura o escritura pendiente. En general, será la última orden que haga referencia al archivo, el que ya no podrá ser utilizado a menos que sea abierto nuevamente. Ver **abrir**.

Ejemplo:
cerrar "diccio

---

cerrartodo (comando, archivos, primit.oem)
Cierra todos los archivos que se encuentren abiertos excluyendo el archivo de copia.

concopia (archivo) (comando, archivos, primit.oem)
Abre un archivo con el nombre indicado por su argumento; toda la información que aparezca a continuación en la pantalla, será copiada en ese archivo (exceptuando los gráficos) hasta que se ordene **sincopia**.

Ejemplo:
```
?concopia "curso
```

---

copiarpantalla (comando, archivos, primit.oem)
Genera una copia por impresora de la pantalla que se está visualizando. No se copian las tortugas a menos que estén estampadas (ver manual Talent MSX LOGO - primitiva **estampar**). Se obtiene el mismo resultado pulsando la tecla <SELECT>.

Ejemplo:
```
si :tecla = "a [copiarpantalla]
```

---

desocupar (comando, música, musica)
Permite guardar las melodías que hemos definido, borrando todos los procedimientos del sistema musical que están en la memoria. Así se evita guardar cada vez los procedimientos del sistema musical y sólo se guardan los que se hayan definido.

---

distancia3d [x y z] (operación, logotri, triaux)
Responde con la distancia, medida en pasos de tortuga, entre la posición actual de la tortuga tridimensional y el punto indicado por el argumento.

Ejemplo:
```
?triesc distancia3d [50 50 50]
86,60254
```

---

escribir.a (objeto) (comando, archivos, primit.oem)
esc.a (objeto) (comando, archivos, primit.oem)
Escribe sus argumentos en un archivo designado previamente con la orden **fesc** y pasa a la línea siguiente.

Ejemplo:
```
?escribir.a :nombre
?esc.a [Hola que tal]
```

escribirs.a (objeto) (comando, archivo, primit.oem)
Escribe sus argumentos en un archivo designado previamente con la orden fesc sin pasar a la línea siguiente.

Ejemplo:

```
?escribirs.a :nombre
?esc.a :telefono
```

(Escribe el teléfono a continuación del nombre y luego pasa a la línea siguiente).

---

extensiones (comando, archivos, primit.oem)
Muestra la lista de las primitivas que se incorporan con el manejo de archivos y la copia de pantalla por impresora. Equivale a **primitivas**.

---

fesc (archivo) (comando, archivos, primit.oem)
Designa el archivo especificado por su argumento, para operaciones de escritura. El archivo tiene que haber sido abierto previamente. Ver **abrir**.

Ejemplo:

```
?fesc "diccio
```

fleer (archivo) (comando, archivos, primit.oem)
Designa el archivo especificado por su argumento, para operaciones de lectura. El archivo tiene que haber sido abierto previamente. Ver **abrir**.

Ejemplo:
?fleer "diccio

---

fpos3d [x y 2] (comando, logotri, triaux)
Mueve la tortuga a la posición especificada por su argumento sin variar su orientación. Si la tortuga está con pluma, dejará un rastro al moverse.

Ejemplo:
?fpos3d [50 50 50]

---

fposesc (número) (comando, archivos, primit.oem)
Determina la posición del archivo designado para escritura a partir de la cual se ejecutará la próxima orden de escritura. La posición se calcula en caracteres.

Ejemplo:
?fposesc 15

---

fposleer (número) (comando, archivos, primit.oem)
Determina la posición del archivo designado para lectura a partir de la cual se leerá el próximo carácter o línea. La posición se calcula en caracteres.

Ejemplo:
?fposleer 20

frumbo3d [[c11 c12 c13][c21 c22 c23][c31 c32 c33]] (comando, logo tri, triaux)

Impone a la tortuga tridimensional el rumbo indicado por su lista argumento; cada sublista de la lista argumento debe estar formada por los tres cosenos directores de cada uno de los ejes que definen la nueva orientación de la tortuga.

Ejemplo:

?frumbo3d [[0 1 0][-1 0 0]7[0 0 1]]

---

hacia3d [x y 2] (operación, logotri, triaux)

Responde con la lista de cosenos directores de los tres ejes que tendría la tortuga si estuviera mirando hacia el lugar especificado por el argumento. El eje longitudinal de la tortuga es determinado por la línea recta que va desde la posición presente de la tortuga a la posición dada. Por convención, el eje transversal es horizontal y el eje perpendicular se dirige hacia arriba.

Ejemplo:

?triesc hacia3d [45 45 45]
[0,57735027 0,57735026 0,57735027] [0,70710678 0 -0,70710678] [0,47140452 0,81649658 -0, 40824829]

---

impresora (tipo) (comando, archivos, primit.oem)

Selecciona el tipo de impresora que se utilizará con el comando copiarpantalla. Los tipos de impresora disponibles son: **epson**, **msx** y **seikosha**.

Esta simbología representa a los tres tipos de impresora que comúnmente se encuentran disponibles para MSX: las "tipo Epson", las de la norma MSX y la Seikosha GP550.

Ejemplo:

?impresora "msx

iniciar.música (comando, música, musica)
Inicializa los procedimientos del sistema musical definiendo las frecuencias de las notas que se van a utilizar.

---

lc.a (operación, archivos, primit.oem)
Significa leer carácter de archivo. Lee un carácter del archivo designado para lectura.

Ejemplo:
```
?si lc.a = "a [adelante 50]
```

---

leerfda? (operación, archivos, primit.oem)
Responde cierto si se ha llegado al fin del archivo designado para lectura. En caso contrario, responde falso.

Ejemplo:
```
?si leerfda? [parar]
```

11.a (operación, archivos, primit.oem)
Significa leer línea de archivo. Lee una línea del archivo designado para lectura y la responde como lista.

Ejemplo:
```
?si11.a = :nombre [esc frase [el telefono es:] 11.a parar]
```

---

música (nombre, velocidad, volumen, lista de notas) (comando, música, musica)
Permite definir las notas de una melodía convirtiendo las notas que se ingresan como dato, en las órdenes sonido conocidas, y creando el procedimiento logo cuyo **nombre** se especifica.

Ejemplo:

```
música "parte1 150 10 [[sol c. 0] [sol sc 0] [la n 0] [sol n 0] [do 5 n 0] [si b 0] ]
```

---

pos3d (operación, logotri, triaux)

Responde con una lista formada por las coordenadas x, y, z de la tortuga.

Ejemplo:

```
?tri andar 30 virar 40 andar 60
?esc pos3d
-38,567256 75,962666 0
```

---

posesc (operación, archivos, primit.oem)

Responde con la posición del archivo designado para escritura a partir de la cual se ejecutará la próxima orden de escritura (medida en cantidad de caracteres).

Ejemplo:

```
?hacer "ubicacion posesc
```

---

posleer (operación, archivos, primit.oem)

Responde con la posición del archivo designado para lectura a partir de la cual se leerá la próxima línea o carácter (medida en cantidad de caracteres).

Ejemplo:

```
?si posleer = 100 [parar]
```

---

rde (número) (comando, logotri, breves)

Significa **rolar** a la derecha. Equivale a rolar.

---

riz (comando, logotri, breves)

Significa **rolar** a la izquierda. Equivale a rolar con argumento negativo.

rolar (número) (comando, logotri, tri)
Gira la tortuga tridimensional según su eje transversal el ángulo indicado por su argumento.

Ejemplo:
?rolar 60

---

rumbo3d (operación, logotri, triaux)
Responde con el rumbo de la tortuga en el espacio como una lista de listas; cada sublista está formada por los tres cosenos directores de cada uno de los ejes de la tortuga.

Ejemplo:
?mostrar rumbo3d
[[0 1 0] [-1 0 0] [0 0 1]]

---

sincopia (comando, archivos, primit.oem)
Cierra el envío de información al archivo creado con la orden concopia. Ver **concopia**.

---

tri (comando, logotri, tri)
Prepara la escena para la realización de dibujos tridimensionales. Borra la pantalla y deja la tortuga lista para dibujar, en estado visible y en modo ventana.

---

triada (comando, logotri, tri)
Crea la triada y hace que aparezca - si no es visible - o desaparezca – si es visible- en la pantalla. La tríada está formada por tres ejes que indican la orientación de la tortuga tridimensional en el espacio.

---

vde (número) (comando, logotri, breves)
Significa virar a la derecha. Equivale a **virar** con argumento negativo.

virar (número) (comando, logotri, tri)
Gira la tortuga tridimensional en su propio plano, el ángulo indicado por su argumento, de una forma similar a las órdenes **derecha** e **izquierda** de la tortuga plana clásica.

Ejemplo:
?virar 90

---

viz (número) (comando, logotri, breves)
Significa virar a la izquierda. Equivale a **virar**

Apéndice C
Primitivas y procedimientos que incorporan las extensiones al Talent MSX LOGO agrupadas por función.

Sistema musical

desocupar
iniciar.música
música

Manejo de archivos y copia de pantalla

abrir
cerrar
cerrartodo
concopia
copiarpantalla
escribir.a
esc.a
escribirs.a
extensiones
fesc
fleer
fposesc
fposleer
impresora
lc.a
leerfda?
11.a
posesc
posleer
sincopia

LOGO en el espacio tridimensional

aad
aat
andar
cabecear
cad
cat

distancia3d
fpos3d
frumbo3d
hacia3d
pos3d
rde
riz
rolar
rumbo3d
al
triada
vde
virar
viz

EXTENSIONES

MSX LOGO

Brindamos una ampliación al MSX-Logo de Talent que incorpora algunas facilidades que el cartucho no posee, y son útiles para dar un máximo aprovechamiento al resto de los periféricos, como por ejemplo la unidad de discos flexibles o impresora.

Las extensiones que se incorporan están agrupadas en 3 tipos de funciones:

a) Manejo de archivos
b) Sistema musical Logo
c) Logo tridimensional

*** Manejo de archivos:**
MSX-Logo provee diversas posibilidades para manejo de diskettera e impresora, mediante la definición de nuevas primitivas.

MSX-Logo puede manejar distintos tipos de archivos, mediante los cuales se pueden almacenar información de programas, datos, etc., en una memoria auxiliar representada por la unidad de discos flexibles y su lugar físico de almacenamiento: el diskette.

La otra opción que ofrece el manejo de archivos está dada por la transcripción de la pantalla a la impresora, copiando los textos y dibujos en el papel.

*** Sistema Musical Logo:**
El sistema básico MSX-Logo permite escribir y ejecutar todo tipo de melodías, usando hasta tres voces simultáneamente, mediante la primitiva sonido.

Para facilitar la escritura de obras complejas, se ha creado un sistema musical con nuevas órdenes.

La orden música acepta como argumentos los nombres de las notas de la escala cromática (do, do#, re, etc.) y las abreviaturas de las figuras rítmicas conocidas (r=redonda; b=blanca; n=negra; etc.) y además permite definir el volumen y la velocidad de un trozo musical completo, sin necesidad de hacerlo para cada nota.

*** Logo Tridimensional:**
Esta extensión del MSX-Logo brinda la posibilidad de manejar coordenadas espaciales en tres dimensiones.

Con el MSX-Logo TRI se pueden crear micromundos tales que el usuario visualice objetos tridimensionales, estudie perspectivas, desarrolle el sentido de la orientación espacial, de forma a objetos en el espacio, etc.

IMPORTANTE

El presente manual incluye un diskette que contiene las Extensiones MSX-Logo de Talent, y para su utilización se requiere además de la computadora personal Talent-MSX, el cartucho del MSX-Logo de Talent y una unidad de disco flexible Talent-MSX.
Estas extensiones fueron desarrolladas exclusivamente para el MSX-Logo de Talent y no funcionan con ningún otro Logo desarrollado para MSX.

Sistema Musical Logo y Logo Tridimensional fueron desarrollados por Fernández Long y Reggini S.A. para TELEMÁTICA S.A.

Producido en Argentina por Telemática S.A., en la Provincia de San Luis